Instituto de Informática - UFRGS

Sistemas Operacionais

Alocação de memória

Aula 18

Introdução

- Constatações sobre o uso de memória
  - Cada processo necessita de um conjunto mínimo de páginas para executar eficientemente
  - O conjunto de páginas necessárias é dinâmica, isto é, evolui com a execução do processo
- O objetivo é manter o conjunto ótimo de páginas de um processo em memória
  - Cada processo tem um comportamento (característica local)
  - Representa um problema de alocação

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

O problema da alocação

- Um processo só executa se estiver carregado na memória RAM
  - Completamente ou parcialmente
- Problema: como alocar memória a processos?
  - Superdimensionamento: baixo grau de multiprogramação
  - Subdimensionamento: leva a ultrapaginação (*thrashing*)
- Questões:
  - Qual o mínimo de quadros necessário por processo?
  - Quanto alocar para cada processo?
  - Caso ocorra falta de páginas, onde obter (alocar) os quadros necessários?
    - E se não houver mais memória disponível?

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

A questão da quantidade mínima de quadros

- Um processo necessita um número mínimo de quadros para executar
  - Definido pelo conjunto de instruções de máquina (modos de endereçamento)
    - e.g: análise por instrução
      - INC MEM necessita duas páginas (código, dados)
      - MOV MEM, POS necessita três páginas (código, 2 de dados)
  - Própria sequência de execução
- Quanto menor o número de quadros alocado a um processo maior a probabilidade de taxa de falta de páginas (*page fault*)
  - Queda de desempenho do sistema
  - Tentar manter o máximo de páginas em memória
- Número máximo de páginas depende da memória física da máquina

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

Alocação de memória em sistemas com paginação

- Alocação estática: todas as páginas tem que estar carregadas na memória para o programa executar
  - Se não há espaço suficiente
    - Processo não é criado
    - Sistema realiza o *swap* em um ou mais processos
- Alocação dinâmica: as páginas são carregadas de acordo com a execução
  - Se não há espaço suficiente
    - Suspende o processo
    - Sistema realiza *swap* em um ou mais processos
    - Libera um quadro ocupado (Política de substituição) - *pagging*

Quanto alocar?

- O problema é determinar quantos quadros serão alocados para cada processo
- Duas estratégias
  - Fixa
    - Alocação igualitária
    - Alocação proporcional
  - Variável
    - Baseado na análise periódica da localidade de referência
    - Vinculado a uma política de substituição global (mais adiante)

Alocação igualitária

- Princípio é dividir os *m* quadros da memória física entre os *n* processos no escalonador de curto prazo
  - Cada processo recebe *m/n* quadros
  - A "sobra" pode compor um *poll* de quadros livres
  - Número de quadros é ajustado dinamicamente em função do grau de multiprogramação
- Provoca distorções já que os processos possuem diferentes necessidades de memória

Alocação proporcional

- Princípio é alocar quadros (*f*) em função do tamanho do processo:

$$f_i = \frac{s_i}{\sum_{j=1}^{n} s_j} \times m$$

s_i: memória virtual do processo p_i
n: número de processos em estado apto
m: número de quadros

- A alocação deve ser reajustada dinamicamente em função do grau de multiprogramação
- Possibilidade de "ponderar" número de quadros com prioridades

Exemplo

- Características da memória
  - Memória disponível: 1GB
  - Tamanho das Páginas: 1kB
- Situação atual de alocação
  - Memória livre para alocação: 62 frames
- Demanda de alocação
  - Processos demandantes: 2
  - Páginas requisitadas pelo processo 1: 10 páginas
  - Páginas requisitadas pelo processo 2: 127 páginas
- Cálculo das páginas alocadas para cada processo
  - Processo 1: (10/137)x62=4
  - Processo 2: (127/137)x62=57

De onde obter quadros na ausência de disponíveis?

- Estratégia global:
  - Memória é vista como conj. de quadros compartilhado por todos processos
  - Em caso de necessidade, qualquer página em um quadro é candidata a ser substituída
    - Independentemente do processo
- Estratégia local:
  - Mantém por processo um conjunto de quadros em uso
  - Em caso de necessidade, seleciona uma página em um quadro pertencente ao processo para ser substituída
- Algoritmos de substituição de páginas
  - Podem ser independentes da estratégia (FIFO, LRU e variantes)
  - São ligados a estratégia (*working set*)

Vantagens e desvantagens

- Global
  - Os processos não são capazes de controlar sua taxa de falta de páginas
    - Alocação depende do comportamento dos outros processos
    - O comportamento de cada processo pode variar significativamente, de uma execução para outra
- Local
  - Os processos têm alocado um número de quadros
    - Impede outros processos de usarem quadros pouco usados
- Método mais comum: global
  - Prioridade para o critério “utilização de memória”
  - Ou “redução do desperdício”

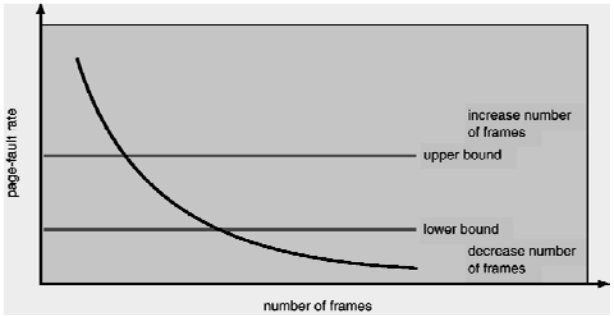
Controle de alocação

- Sistema para realimentar o alocação de quadros
  - Working set
  - Frequência de falta de páginas
- Working set
  - Identifica qual o conjunto de páginas é necessário para uma certa localidade de referência
    - Cardinalidade informa a quantidade de páginas
    - Elementos indicam quais páginas são necessárias
  - Útil para pré-paginação e *swap-in* de processos
- Frequência de falta de páginas indica quantas páginas são “confortáveis” para um processo
  - Não indica quais substituir

Método de frequência de falta de páginas (FFP)

- A cada falta de páginas avalia o conjunto de páginas residentes em memória
  - Remove da memória todas as páginas não referenciadas durante $t_i - t_{i-1} > \tau$ onde t_i é o instante de tempo do acesso em que ocorre a falta de página
  - Carrega na memória a página faltante
- Efeito prático é o estabelecimento de uma taxa de falta de páginas
  - Se a taxa é alta, processo necessita de mais frames
  - Se a taxa é baixa, processo libera frames

Método da frequência de falta de páginas



Escolhendo o processo "vítima" para *swap-out*

- Critérios possíveis:
  - O processo de menor prioridade
  - O processo que provocou a falta de página
  - O último processo que executou
  - O menor processo
  - O maior processo
- O melhor critério é dependente do sistema e de sua aplicação

Pré-paginação

- Consiste em trazer para a memória todo o *working-set* de um processo
  - Quando processos realizam transições dos suspenso para apto/bloqueado
    - Escalonamento de médio prazo
- Pré-paginação versus fala de páginas
  - Quantas páginas dentre as pré-paginadas serão efetivamente usadas?
  - Custo da pré-paginação deve ser menor que custo de tratamento de falta de páginas
    - Vale a pena: se o custo de trazer (1- α) páginas é menor que o tratamento de falta de α páginas.

Efetividade da Pré-paginação

- Parâmetros
  - PP = custo para pré-paginar uma página
  - FP = custo de tratamento de uma falta de página
  - N = número de páginas a serem pré-paginadas
  - a = percentual de página efetivamente usadas, dentre as pré-paginadas
- Custos
  - Custo de pré-paginação de N páginas
    - Cpp = N x PP
  - Custo de utilização por faltas de paginas
    - Cfp = a x N x FP

Comparação de Custos

- Vale a pena usar pré-paginação se Cpp < Cfp

$$a > \frac{PP}{FP}$$

  - N x PP < a x N x FP
  - PP < a x FP
- Desperdício de recursos em cada mecanismo
  - No caso de PP
    - O desperdício é devido ao pré-paginar ((1-a) x N) páginas
  - No caso de FP
    - O desperdício é devido ao tratamento das faltas de (a x N) páginas

Fatores adicionais

- Novas arquiteturas (NUMA)
- Alocação de memória para o núcleo
- Páginas para E/S (*pinning* – já visto)
- Arquivos mapeados em memória

Máquinas NUMA

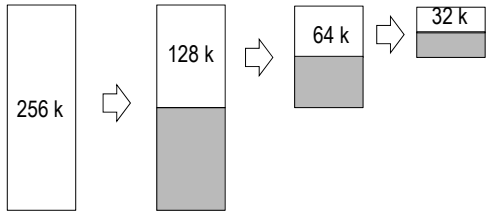
- *Non-Uniform Memory Access* (NUMA)
  - Composta por um conjunto de módulos de CPUs e memória
    - Módulos (nós NUMA) são interligados por uma topologia
  - Fator NUMA: relação entre o acesso remoto e acesso local
- Alocação de quadros
  - Deve considerar a forma como a memória está conectada à(s) CPU(s)
  - Considerar uma alocação uniforme, implica em baixo desempenho

Alocação de memória para o núcleo

- Necessidades diferentes dos processos de usuários:
  - Alocação de memória para estruturas de dados de tamanho variável
    - Muito dinâmico na criação e destruição dessas estruturas (e.g. PCB)
    - Paginação implica em alocar ao menos uma página → frag. interna
  - Necessidade de áreas contíguas para determinadas operações de E/S
- Solução:
  - Alocação de memória em área específica (pré-reservada)
  - Algoritmos de alocação diferenciados
    - Buddy
    - Slab

Sistema *buddy*

- Compromisso entre partição fixa e partição variável
- Considera um segmento de memória
- Memória é dividida em um certo número de blocos de espaços livre
  - Tamanho em potência de 2
  - Um bloco de tamanho 2^i é dividido em 2 blocos de tamanho 2^{i-1} (*buddies*)

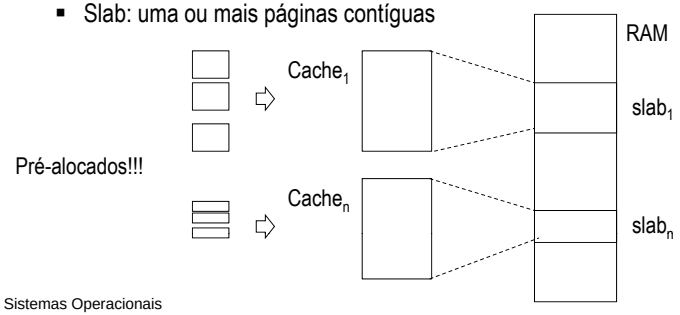


Sistema *buddy:* vantagens e desvantagens

- Vantagem:
  - Rapidez na alocação
  - Capacidade de fusionar buddies
- Desvantagem:
  - "Arredondar" para a próxima potência de 2 leva a fragmentação interna
- Custo "gerencial"
  - Necessário manter a lista de blocos livres e ocupados, concatenar buddies etc
- Exemplo de emprego:
  - Operações de E/S transferem diretamente dados de periféricos para memória via DMA, curto-circuitando a gerência de memória
    - Solução é alocar contiguamente uma área de memória para E/S

Alocação slab

- Baseado em uma hierarquia de componentes
  - Objeto – cache – slab
- Princípio:
  - Objetos são estruturas genéricas próprias do núcleo (PCBs,IOBlock,FCB, etc)
  - Uma cache para cada "classe" de objetos
  - Slab: uma ou mais páginas contíguas



Arquivos mapeados em memória

- Acesso a um arquivo:
  - open( ) – read ( )/write( ) – close( )
  - Exige uma chamada de sistema e um acesso a disco para cada operação
- Mapeamento do arquivo no espaço de endereçamento
  - Comportamento similar à paginação por demanda
    - Blocos do arquivo são transferidos para páginas
    - Escrita no momento em que o mapeamento é desfeito
  - Arquivo funciona como "área de swap particular" para aquele trecho de espaço de endereçamento virtual
- Acesso a um arquivo mapeado em memória
  - mmap( ) – acessos normais a memória via ponteiro → ummap( )

Leituras complementares

- A. Tanenbaum. *Sistemas Operacionais Modernos* (3ª edição), Pearson Brasil, 2010.
  - Capítulo 9: seção 3.5
- A. Silberchatz, P. Galvin; *Sistemas Operacionais*. (7ª edição). Campus, 2008.
  - Capítulo 9: seções 9.5 a 9.9
- R. Oliveira, A. Carissimi, S. Toscani; *Sistemas Operacionais*. Editora Bookman 4ª edição, 2010
  - Capítulo 6 e capítulo 7