MAC0422 - Sistemas Operacionais

Daniel Macêdo Batista

IME - USP, 23 de Novembro de 2020

Roteiro

Algoritmos de substituição de páginas

Algoritmos de substituição de páginas

Por quê?

- Quando ocorre falha de página, o SO tem que escolher uma página para remover da memória
- Se a página foi modificada enquanto ela estava na memória, ela precisa ser reescrita para o disco para que o conteúdo esteja atualizado
- Se a página não foi modificada (se só contém código por exemplo), não é necessário reescrever, basta sobrescrever a página com o novo conteúdo
- Qual página é a melhor opção para ser sobrescrita?
  - Aleatório?
  - Página pouco usada (Evitar páginas muito usadas)
  - Problema similar na memória cache e servidores web
  - Página do próprio processo ou de outros processos?

Ótimo

- Assim que uma falha de página ocorre, algumas outras páginas estão na memória principal, 1 delas será referenciada na próxima instrução, outras só daqui a 10, 100, 1000 instruções adiante
- Suponha que é possível rotular cada página com a quantidade de instruções que serão executadas antes daquela página ser acessada
- De posse dos rótulos, remova a página com o maior rótulo
- **Fácil de descrever** mas **impossível de implementar**

Not Recently Used Page

- Suponha (isso existe em muitos computadores com suporte à memória virtual) que cada página possui 2 bits associados:
  - `R` é 1 se a página é referenciada (lida ou escrita)
  - `M` é 1 se a página é modificada (escrita)
- Os bits são atualizados em qualquer referência à memória
- Os bits só são zerados pelo SO
- Quando um processo é inicializado, `R` e `M` valem zero
- De tempos em tempos o bit R é modificado para zero

Not Recently Used Page

- Quando ocorre uma falha
  - O SO divide os processos em 4 classes e escolhe uma página dessas classes (das menores para as maiores) para substituir. Mais de uma página dentro da classe escolhida $\rightarrow$ Escolhe qualquer uma delas
  - Classe 0: R=0, M=0
  - Classe 1: R=0, M=1
  - Classe 2: R=1, M=0
  - Classe 3: R=1, M=1

FIFO

- O SO mantém uma lista de todas as páginas que estão na memória física **ordenadas pelo instante de "chegada" de cada página**
- Por exemplo: no início da lista, a página mais velha e no fim da lista, vai anexando as páginas mais novas
- Quando ocorre uma falha de página:
  - A página no início da lista é removida e o quadro de página dela é usado para a nova página (lembrando da cópia para o disco se necessária)
  - Os bits e os mapeamentos são atualizados
  - A nova página é adicionada no fim da lista
- **Um algoritmo de baixo overhead mas não trata páginas muito acessadas por isso é raramente utilizado**

Segunda chance

- **Tentativa de melhorar o FIFO sem perder o baixo overhead**
- Quando ocorre uma falha de página:
  - Se o bit R da página no início da lista é 0, age como no FIFO
  - Se o bit R da página no início da lista é 1, o bit é alterado para 0, esta página é movida para o fim da lista e o processo repete para a nova página do início da lista
- A ideia é buscar uma página velha que não tenha sido acessada recentemente
- **Se todas as páginas tiverem o bit R = 1, fica pior que o FIFO em termos de overhead**
- **Outro problema do Segunda Chance é ficar manipulando, talvez demais, a lista de páginas**

Relógio

- **Tentativa de reduzir o overhead do Segunda Chance causado quando há muita manipulação na lista de páginas**
- A ideia é usar uma lista circular e um ponteiro apontando para a página mais velha
- Quando ocorre uma falha de página:
  - Se o bit R da página apontada pelo ponteiro é 0, age parecido com o FIFO com a diferença de que o ponteiro avança 1 posição
  - Se o bit R da página apontada pelo ponteiro é 1, o bit é alterado para 0, avança o ponteiro 1 posição e repete o procedimento para a nova página apontada (Repete até encontrar uma página com o bit R igual a 0)

Menos usada recentemente

- *Least Recently Used (LRU)*
- O que faz mais sentido é ter algo próximo do ótimo (remover a página que vai ser acessada o mais tarde possível)
- **O máximo que dá para fazer é inferir que páginas muito acessadas no passado recente, serão no futuro recente (o contrário também vale)**
- A ideia aqui é ter algo mais próximo do ótimo do que o NRU (lá no NRU sabe-se que a página foi lida ou escrita mas a informação de instante de tempo não existe)
- Quando ocorre uma falha de página: remova a página que não tenha sido acessada pelo maior intervalo de tempo

Menos usada recentemente

- **Requisito (difícil de ser implementado com baixo overhead):** lista de todas as páginas na memória física com a página mais recentemente usada no início e a página menos recentemente usada no fim, além de ter que manipular a lista
- Mesmo a manipulação da lista podendo ser melhorada com uma lista circular, manter a informação dos acessos das páginas é custoso a cada acesso
- Pensando numa implementação... (várias)

Menos usada recentemente – Primeira versão

- Considere um contador de 64 bits incrementado automaticamente após a execução de 1 instrução
- Adicione em cada entrada na tabela de páginas um campo no qual caiba o valor desse contador
- Após cada referência à memória, o valor atual do contador é armazenado na entrada da tabela de página referenciada
- (Relembrando) Quando ocorre uma falha de página: remova a página que não tenha sido acessada pelo maior intervalo de tempo
- (Agora) **Quando ocorre uma falha de página: remova a página com o menor valor no campo do contador**

Menos usada recentemente – Segunda versão

- Considere uma matriz de n X n bits, onde n é a quantidade de quadros de páginas (tamanho da memória física)
- Inicialize todas as entradas da matriz com 0
- Após cada referência à memória no quadro de página k, todos os bits da linha k na matriz são alterados para 1 e todos os bits da coluna k para 0.
- (Relembrando) Quando ocorre uma falha de página: remova a página que não tenha sido acessada pelo maior intervalo de tempo
- (Agora) **Quando ocorre uma falha de página: remova a página p cujo valor binário da linha na matriz seja o menor**

Menos usada recentemente – Segunda versão

- Exemplo: Quatro quadros de página e acessos: 0 1 2 3 2 1 0 3 2 3
- Conteúdo final da matriz (a página a ser removida seria a 1):

0	1	0	0
0	0	0	0
1	1	0	0
1	1	1	0

Menos usada recentemente – Terceira versão

- Considere um contador em software associado a cada página (inicialmente valendo 0)
- A cada pulso de clock, o SO olha todas as páginas na memória física e adiciona ao contador o valor atual do bit R (continua zerando R regularmente)
- (Relembrando) Quando ocorre uma falha de página: remova a página que não tenha sido acessada pelo maior intervalo de tempo
- (Agora) **Quando ocorre uma falha de página: remova a página p cujo valor do contador seja o menor**
- **Problema dessa versão:** Nunca esquece de nada! Páginas muitíssimo acessadas por um intervalo de tempo curto dificilmente serão removidas!