MAC0422 - Sistemas Operacionais

Daniel Macêdo Batista

IME - USP, 26 de Novembro de 2020

Roteiro

Algoritmos de substituição de páginas

Algoritmos de substituição de páginas

Menos usada recentemente – Quarta versão

- No fim das contas o algoritmo da terceira versão está substituindo a página não usada com frequência (*Not Frequently Used* – NFU)
- Modificando a terceira versão (tentando achar um jeito de incluir tempo – envelhecer os contadores):
  - contadores são deslocados para direita 1 bit antes do R ser adicionado neles
  - o bit R é adicionado ao bit mais significativo do contador e não ao menos significativo

Menos usada recentemente – Quarta versão

- A ideia é simular uma idade para os acessos
- (Relembrando) Quando ocorre uma falha de página: remova a página que não tenha sido acessada pelo maior intervalo de tempo
- (Agora, continua) **Quando ocorre uma falha de página: remova a página p cujo valor do contador seja o menor**

Menos usada recentemente – Quarta versão

- Exemplo (no início): 6 páginas, com bits R valendo 1, 0, 1, 0, 1 e 1. Considere um contador de 8 bits
- Clock 1: acessos às páginas 0, 1 e 4
- Clock 2: acessos às páginas 0, 1, 3 e 5
- Clock 3: acessos às páginas 0 e 4
- Clock 4: acessos às páginas 1 e 2
- Contadores no final por página: 01111000, 10110000, 10001000, 00100000, 01011000, 00101000 (Removeria a página 3)
- **Vantagem:** tem uma visão do passado (deslocando os bits) mas não mantém essa visão por muito tempo (o contador tem limite)

Conjunto de trabalho

- *Working set*
- **Lembrando** que estamos considerando que nenhuma página dos processos começa na memória física → De cara um monte de page fault acontece mas estabiliza
- Essa estratégia de carregar as páginas quando precisa é chamada de paginação sob demanda
- **Lembrando** que há localidade espacial
- As páginas que um processo está usando atualmente (em loops por exemplo) são chamadas de conjunto de trabalho

Conjunto de trabalho

- Quando um conjunto de trabalho está inteiro na memória física, não há falhas de página, até que vá para outra parte do código (outro loop por exemplo)
- Se a memória física livre não é suficiente para manter um conjunto de trabalho inteiro $\rightarrow$ Muitas falhas de páginas $\rightarrow$ *trashing*

Conjunto de trabalho

- Outro ponto importante: o que fazer quando um processo ganha de novo a CPU (escalonamento)? Da forma como visto até agora, espera voltar a dar falhas de página para carregar de novo as páginas dele na memória
- Esse processo vai deixar o SO muito ineficiente
- Como melhorar?
  1. Conhecer os conjuntos de trabalho dos processos
  2. Fazer pré-paginação (carregar o conjunto de trabalho antes do processo começar a rodar)

Conjunto de trabalho

- Deixando mais claro, um conjunto de trabalho $w(k, t)$ define que a qualquer instante de tempo t, existe um conjunto w de todas as páginas usadas pelas k mais recentes referências de memória
- a função w em função de k começa "rápido" com um limite finito

Conjunto de trabalho – Ideia do algoritmo

- Como implementar? O SO precisa manter registro de quais páginas estão em um conjunto de trabalho
- Quando ocorre uma falha de página: remova qualquer página que não esteja no conjunto de trabalho de algum processo em execução
- Como manter o conjunto de trabalho atualizado? (Precisa definir k)
- Conhecendo o valor de k: a cada referência à memória atualiza os k mais recentes acessos

Conjunto de trabalho – primeira versão

- Ter um registrador de deslocamento (similar ao contador da quarta versão do LRU)
- Cada posição do registrador armazena um endereço de página
- Tamanho do registrador: k
- A cada referência à memória: desloca o registrador 1 posição para a esquerda (perde o acesso mais velho)
- A cada referência à memória: põe na posição mais à direita o endereço da página acessada
- O conteúdo do registrador é o *working set*
- Quando precisar substituir uma página, substitui alguma que não esteja em nenhum dos registradores
- **Problema**: alto *overhead*

Conjunto de trabalho – segunda versão

- Ao invés de contar os k últimos acessos, considerar intervalo de tempo
- Por exemplo: manter registro das páginas acessadas nos últimos 100ms (tempo τ)
- Importante observar que cada processo teria o seu próprio registro e consideraria apenas o tempo de CPU que ele de fato ganhou (não conta o tempo em que ficou no estado de "Pronto" ou "Bloqueado")
- Na prática para cada página precisa manter: **instante de tempo** em que a página foi acessada pela última vez e o bit **R**

Conjunto de trabalho – segunda versão


```
for (cada pagina) {
  if (R==1)
     tempo = t virtual do processo;
  elif (R==0 && (t virtual do processo - tempo) > tau)
     remova esta pagina;
  elif (R==0 && (t virtual do processo - tempo) <= tau)
     lembra da pagina com o menor tempo;
}
if (todos os bits R sao 1)
  remove alguma pagina aleatoriamente;
elif (nenhuma pagina foi removida)
  remove a pagina com menor tempo de ultimo acesso;
```

Obs.: bits R e M continuam sendo manipulados da mesma forma de sempre