Sumário

1 Memória virtual - Desempenho (aula 27/05)

1.0.1 Otimização

Se encher a memória, é necessária uma política de reposição de páginas. A página só é escrita no disco caso já se tenha escrito nele.

1.0.2 Frames

- Frame: página física
- Page: página virtual

Lugares na memória principal onde colocar as pages.
Dado uma sequênca de acessos, como o algoritmo se comporta?

1. Aleatoriamente;
2. Monitoração de execução de programas reais

1.0.3 FIFO

Retira-se a página que está a mais tempo na memória. Anomalia de **belady**.

1.1 Fifo e segunda chance

1. Segunda chance: FIFO modificado. Se o bit de referência for 0, remova-a. Se for 1, dê uma segunda chance. Fácil de implementar, degenera se todos os bits estiverem setados. Percorre a lista de todas, não pega a mais recente. Estão todos setados caso a memória esteja com muita demanda.
2. Segunda chance ao segunda chance: páginas na ordem de 1 a 4. De sem ref e limpa a com ref e suja.

1.1.1 Política ótima

Não factível! Pode ser aproximada. Número de page faults é menor.

1.1.2 LRU

Remove a página que foi usada há mais tempo. Implementação complicada, pois tem que guardar cada posição de memória e seu tempo! Não sofre da anomalia de Belady!

1. Aproximações
   (a) **LRU:** Basta inicializar o bit de referência com 0 e deixar o hardware colocá-lo em 1 quando a página for acessada. E quando todas as páginas tiverem sido referenciadas?
   (b) **LRU:** mais bits de referência: Guardar o número de bits em um byte. SHR e &. A página com o menor valor do byte de referência deve ser removida.

1.1.3 LFU e MFU = implementações caras.

1.2 Modelo de working set

Janela de trabalho é uma medida de localidade. Δ é seu tamanho. É a medida das últimas delta references a páginas da memória.

O tamanho médio da janela é o número de frames a serem alocados por processo.

1.2.1 Como definir Δ?

Monitoração de um conjunto de programas.

1. Formalmente Não permite a utilização de mais que n frames.
2. **Informalmente** (melhor jeito) Assume-se que o processo vai usar n frames, mas não se garante. Serve para ter uma ideia de quantos processos o sistema consegue tratar de forma eficiente.

1.3 Thrasing

Solução para o caso de haverem mais processos.

Número de page faults aumenta consideravelmente. Quando o pc para e não responde mais (gasta-se mais tempo decidindo o que fazer, do que fazendo algo). **Ex:** ineficência de plugins de navegadores.

2 Memória virtual - Linux (aula 29/05)

Compartilhamento de memória permite um ganho significativo no desempenho.

No Linux podemos identificar compartilhamento de memória:

- Entre buffers de bloco e página;
- Entre páginas através de copy-on-write;
- Entre subrotinas sendo executadas.

Foco na memória principal.

2.1 Entre buffers

2.1.1 Buddy heap

Páginas são agrupadas em blocos de dois por listas encadeadas.

2.1.2 Contadores de referência

Usados para saber se alguma página já foi usada e pode ser liberada ou se ainda existe alguém que precise dela.

Inicia-se com c=0. Quando cria-se uma página, c++; quando é destruída, c−−.

Simples e eficiente.

2.2 Entre páginas

2.2.1 Diversos sabores de páginas

- Sem pistolão: zero filled (primeiro acesso retorna página vazia).
- Com pistolão: backed (acessos à página retornam páginas de arquivo padrinho).
- Private: acesso por único processo.
- Shared: acesso por vários processos.

2.2.2 Fork

Copia todas as páginas na memória para a nova tarefa. Ineficiente e problemas de acesso.

1. Copy-on-write (quase máximo compartilhamento) Compartilha a página somente na leitura.

2.3 Subrotinas

2.3.1 a.out

Código todo é linkado automaticamente.

2.3.2 ELF (mais moderno)

Linkado dinamicamente (carrega subrotinas sob demanda). **Shared libs:** se a subrotina estiver na memória, o SO não carrega outra cópia.

3 Arquivos (aula 03/06)

3.1 Definição

Conjunto de dados normalmente não voláteis; menor unidade de informação secundária. Qualquer dado não-volátil tem que ser guardado em arquivos.

3.2 Definido por

- Nome
- Tipo
- Lugar
- Tamanho
- Permissões
- Tempos

3.3 O que mais além de arquivos?

- Diretórios: coleção de arquivos.
- Partições: pedaços de memória secundária

3.4 Operações com arquivos

Criar, ler, mover, reposicionar(bem útil), truncar (cortar de um ponto para frente), apagar.

3.5 Abir/fechar

Não é essencial, apenas uma otimização através de buffers na memória principal. Consegue-se fazer leitura sem abrir o arquivo.

3.5.1 Problemas

- Esquecer de fechar (memory leak)
- Acessos concorrentes

3.5.2 Tomar cuidado com

- Localização no disco (pode estar mudando)
- Reference count pra ter certeza de que ninguém vai apagar coisas que não estão sendo usadas.

3.6 Métodos de acesso

- Sequencial: ler, escrever e rebobinar (originado de fita). Ex: pdf, áudio, zip, tar, .o.
- Direto: ler, escrever, reposicionar (seek) - originado de disco. Ex: banco de dados, binário, libraries, ávores B.

3.7 Diretórios

Estrutura é um DAG (grafo acíclico direcionado)

- Cria arquivo
- Apaga
- Busca
- Lista conteúdo

Não é muito diferente de arquivos. Symlink(shortcut) e hard link().
Append só serve para log.

4 Arquivos - Implementação (aula 03/06)

4.1 Alocação de espaço em disco

4.1.1 Alocação contígua

Fácil implementação e rápido acesso. Não é muito usado.

4.1.2 Encadeadas

Ex: sistema pick - lista duplamente encadeada (só banco de dados).

- Sem fragmentação, simples de aumentar, criar e apagar. Aproveita o máximo o disco.
- Ruim para acesso sequencial. Se um link se perder, perde-se o arquivo todo (corrompe a informação).

  Não é muito usado.

1. FAT Diminui os erros, menos eficiente.

4.1.3 Indexada

Uma fat para todos os arquivos; um índice por arquivo. Não importa a eficiência.

- Tem que usar um índice, mesmo que o arquivo use um bloco só.
- Tamanho ideal é um bloco de disco. Solução: unix *inode*.

5 Sistemas de E/S (aula 05/06)

- mouse é lento
- discos e impressora rápidos
- trânsito de dados complexo, velocidade difernetes: E/S complexos
- Sistema embutido é mais simples.
- Sistemas embutidos são usados em robôs, máquinas, controladores de aeronaves.

5.1 Tipos de dispositivos e/s

Só há duas classificações:

5.1.1 Bloco

- podem ler escrever em qualquer bloco
- ex: discos

5.1.2 Caracteres

- dados são sequência de caracteres
- não tem seek
- ex: termiais, impressoras, mice, clock, video, fitas.

5.2 Componentes de um dispositivo

- Dispositivo
- Controlador

5.2.1 Software que controla (device driver)

Nem sempre funciona: *ioctl*()

5.3 Processador

É feito um mapeamento do processador para os registradores:

- através de comandos especiais de leituras de portas;
- através de mapeamento de memória (mais usado, porém pode levar a erros). Ex associar um endereço ao disco rígido;

5.4 Modo de acesso

Usa-se todos, não consegue-se usar DMA para todos (restrito). Usa-se as interrupções rapidamente, aí tem que usar polling.

5.4.1 Polling

Handshake: fica esperando o resultado chegar (já chegou? já chegou?). Espera ocupada.

5.4.2 Interrupções

Sem espera ocupada. Cpu programa E/S. Quando os dados ficam prontos, o cpu dá um jump (interrupção) e dados são lidos.

5.4.3 DMA (método mais eficiente)

Métodos anteriores possuem transferência de poucos bytes por vez. Vc tem um controlador ligado ao barramento. Está executando um programa, aloca um buffer e manda uma msg pro DMA. DMA guarda as palavras no buffer e gera apenas uma interrupção para voltar. Copia um bloco inteiro por vez.

5.5 Software

Objetivo: ser independente do dispositivo.
Interface única. Tratado como arquivo.

- Blocante ou assíncrono (em geral é)
- Não blocante

Programas são síncronos.

- Compartilhar (ex: discos)
- Não compartilhar (ex: impressoras)

6 Segurança (aula 10/06)

6.1 Razões de perda de dados

- Catástrofe naturais
- Erros de hw/sw
- Erros humanos
- Erros bizantinos (invasores)

6.2 Invasores

- Passivos: leitura
- Ativos: modificação
- Ex: casuais, internos (pertence à organização), externos (), determinados, hackers, ladrões, espionagem, violação de privacidade.