ARMAZENAMENTO SECUNDÁRIO, PARTE 2

Professora Rosane Minghim

2011

Baseado no materiais de Leandro C. Cintra e M.C.F. de Oliveira

Fonte: Folk & Zoelick, File Structures

Disco como gargalo

- Discos são muito mais lentos que as redes ou a CPU

- Muitos processos são “disk-bound”, i.e, CPU e rede têm que esperar pelos dados do disco

Técnicas p/ minimizar o problema

Multiprogramação: CPU trabalha em outro processo enquanto aguarda o disco

Stripping: o arquivo é repartido entre vários *drives* (paralelismo)

RAID (*redundat array of inexpensive disks*): http://linas.org/linux/raid.html

Técnicas p/ minimizar o problema

Disk cache: blocos de memória RAM configurados para conter páginas de dados do disco. Ao ler dados de um arquivo, o cache é verificado primeiro. Se a informação desejada não é encontrada, um acesso ao disco é realizado, e o novo conteúdo é carregado no cache.

RAM Disk: simula em memória o comportamento do disco mecânico

Fitas Magnéticas

- Fitas: permitem acesso seqüencial muito rápido, mas não permitem acesso direto
- Compactas, resistentes, fáceis de transportar, mais baratas que discos
- Usadas como memória terciária (back-up, arquivo-morto, ...)

Organização dos dados na fita

- Posição de um registro é dada por um deslocamento em bytes (*offset*) relativo ao início do arquivo

- Posição lógica de um byte no arquivo corresponde diretamente à sua posição física relativa ao início do arquivo

Superfície da fita

- A superfície pode ser 'vista' como um conjunto de trilhas paralelas, cada qual sendo uma seqüência bits.

- 9 trilhas paralelas (1 frame): 1 byte + paridade (em geral, paridade ímpar, i.e., o número de bits = 1 é ímpar)

- 1 frame = 1 byte (8 bits em 8 trilhas) + paridade

Superfície da fita

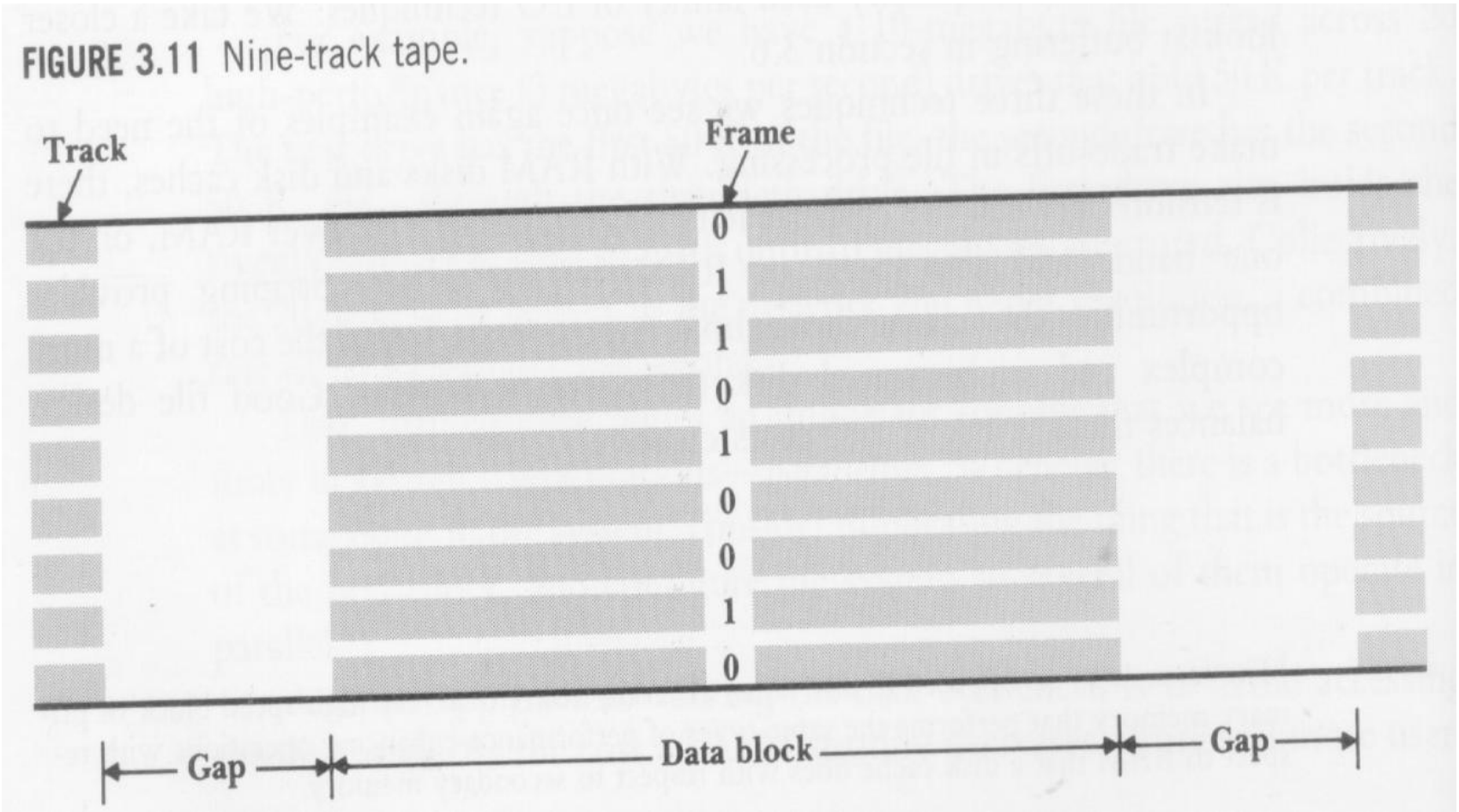


FIGURE 3.11 Nine-track tape.

Superfície da fita

- Frames são agrupados em blocos de dados de tamanhos variados, os quais são separados por intervalos (*interblock gaps*) sem informações

- Intervalos são necessários para viabilizar parada/reinício

Medidas de comparação

- Densidade: bpi - *bytes per inch*
  - Ex: 6.250 bpi
- Velocidade: ips - *inches per second*
  - Ex: 200 ips
- Tamanho do 'interblock gap': *inches*
  - Ex: 0.3 *inches*
- 1 *inch* (polegada) ~ 2,5 cm.

Estimativa do tamanho de fita necessário

- EX: armazenar em fita 1.000.000 de registros com 100 bytes cada. Suponha fita com 6.250 bpi, com intervalo entre blocos de 0.3 polegadas. Quanto de fita é necessário? Sejam:

- b = comprimento físico do bloco de dados (pol.)
- g = comprimento físico do intervalo (pol.)
- n = número de blocos de dados
- S = comprimento de fita necessário (espaço físico) é dado por: S=n*(b+g)

Estimativa do tamanho de fita necessário

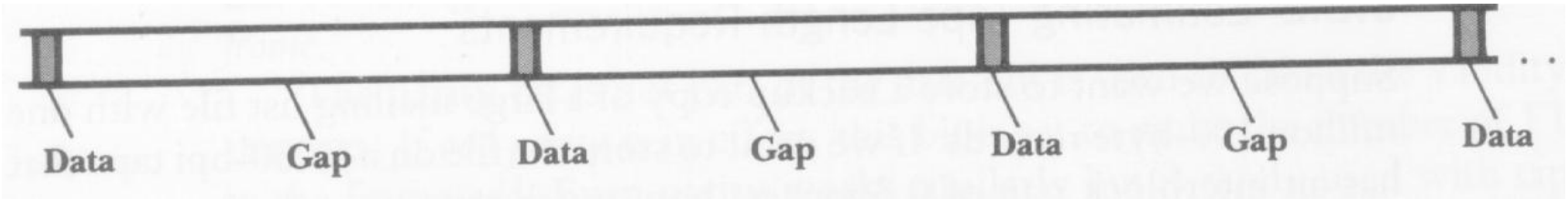
- Supondo 1 bloco = 1 registro:
  S=1.000.000*(100/6.250+0.3)
  S=316.000 pol ~ 7.900 m

- Supondo 1 bloco=50 registros
  - n=1.000.000/50=20.000 blocos
  - b=5000/6250 ~ 0.8 pol
  - S=20.000*(0.8+0.3)=22.000 pol ~ 492 m

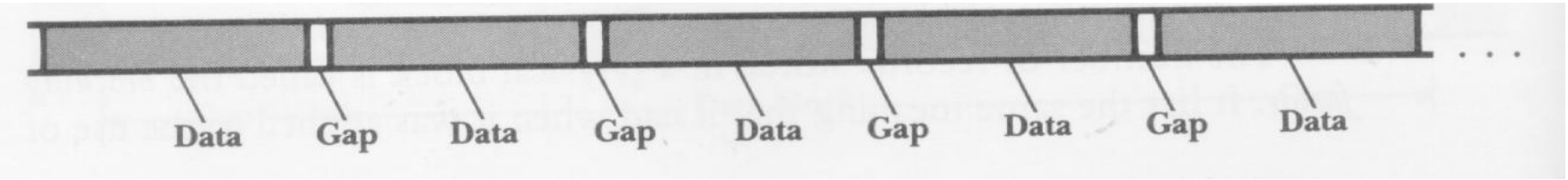
- Comprimentos típicos de fitas: 91 a 1.000 m

Estimativa do tamanho de fita necessário

- 1 registro por bloco



- 50 registros por bloco



Estimativa de tempos de transmissão

- Taxa nominal de transmissão de dados=densidade (bpi)*velocidade (ips)

- Ex: Fita de 6.250 bpi e 200 ips

  taxa transmissão = 6250*200=1.250 KB/s
- Não parece muito ruim, mas... não é a taxa efetiva!

  Porque?

Jornada de um byte

- O que acontece quando 1 programa escreve um byte p/ um arquivo em disco?

Write(arq,&c,1)

Jornada de um byte

- Operações na memória
  - O comando ativa o S.O (file manager), que supervisiona a operação:
    - Verifica se o arquivo existe; se tem permissão de escrita, etc.
    - Obtém a localização do arquivo físico (drive, cilindro, cluster ou extent) correspondente ao arquivo lógico
    - Determina em que setor escrever o byte. Verifica se esse setor já está no buffer de E/S (se não estiver, carrega-o...)

Jornada de um byte

- Operações fora da memória
  - Processador de E/S
    - aguarda a disponibilidade do recurso p/ poder efetivamente disparar a escrita no disco
  - Controlador de disco
    - verifica se *drive* está disponível p/ escrita
    - instrui *drive* p/ mover cabeça de L/E para trilha/setor corretos
    - Disco rotaciona, o setor (e o novo byte) é escrito

Jornada de um byte

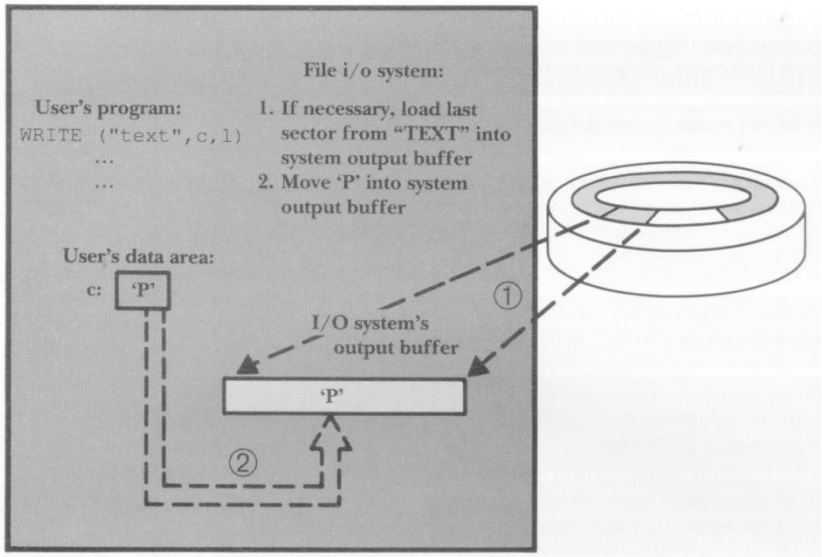


FIGURE 3.15 The file manager moves *P* from the program's data area to a system output buffer, where it may join other bytes headed for the same place on the disk. If necessary, the file manager may have to load the corresponding sector from the disk into the system output buffer.

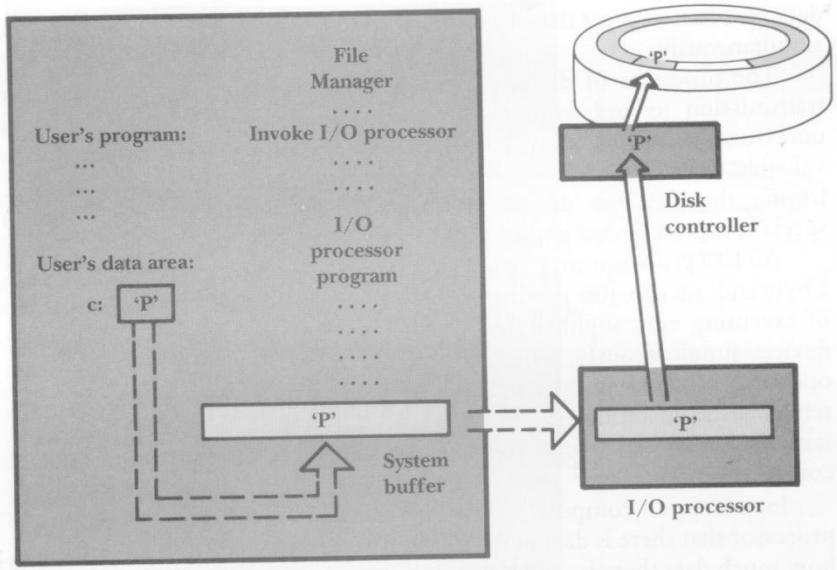


FIGURE 3.16 The file manager sends the I/O processor instructions in the form of an I/O processor program. The I/O processor gets the data from the system buffer, prepares it for storing on the disk, and then sends it to the disk controller, which deposits it on the surface of the disk.

Gerenciamento de buffer

- *Buffering*: permite trabalhar com grandes quantidades de RAM para armazenar informação sendo transferida, de modo a reduzir o nº de acessos ao dispositivo de memória secundária

Buffer como gargalo

Suponha um sistema que utilize um único *buffer*. Em um programa que realiza, intercaladamente operações de leitura/escrita o desempenho seria muito ruim (Porque?).

Os sistemas precisam de, no mínimo, 2 *buffers*: 1 p/ entrada, 1 p/ saída

Buffer como gargalo

- Mesmo com 2 *buffers*, mover dados de e para o disco é muito lento, e os programas podem ficar 'I/O bound'
- Para reduzir o problema:
  - Multiple buffering
    - Double buffering
    - Buffer pooling

FIM