NOVA FACULDADE DE CIÊNCIAS E TECNOLOGIA DEPARTAMENTO DE INFORMÁTICA

Fundamentos de Sistemas de Operação

LEI - 2023/2024

Vitor Duarte
Mª. Cecília Gomes

1

Aula 18

- Cache e buffer de blocos.
- Escalonamento de IO
- Gestão dos blocos do volume
  - inodes, ficheiros e diretorias
  - FFS
- OSTEP: cap. 37, 40, 41

2

Otimizações nos SF

- Fazer *buffering* de blocos durante leituras e escritas
  - Permite leituras/escritas de qualquer dimensão pelos processos sem repetir acessos a disco
  - Melhora as taxas de transferência com blocos maiores ou blocos contíguos
  - Permite *read-ahead* e *delayed-write*
  - Permite escalonar a ordem das leituras e escritas em cada disco
- Fazer cache de blocos de disco
  - Não remover os blocos enquanto a memória não é necessária
  - Futuros acessos e mmap, mesmo de processos distintos, beneficiam do que já está em cache
  - Usar politicas tipo LRU na gestão dessa cache (cache dos discos)

Escritas com buffer/cache de blocos

- Vantagens do *delayed write*
  - Existe uma probabilidade elevada que processos escrevam no mesmo bloco em breve
  - Existe uma probabilidade elevada que se escreva em blocos contíguos
  - Pode otimizar as escritas e manter os blocos do ficheiro também contíguos no disco
- Desvantagens do *delayed write*
  - O sistema de ficheiros em disco pode estar incoerente com as últimas alterações apenas em memória
    - pode perder alterações ou deixar o sistema incoerente
  - Mitigação:
    - escrever quando o ficheiro é fechado
    - Despejar (flush ou sync) periódico dos blocos alterados para o disco (por exemplo de 5 em 5 segundos)

Gestão de blocos e gestão de páginas

- Usar parte da memória para a cache/buffers do IO:
  - Tamanho fixo – Que tamanho? Pode ser demais ou de menos, dependendo do instante
  - Tamanho dinâmico - Compete pela memória com o sistema de memória virtual – gestão mais complicada e pode repetir conteúdo entre os dois
- Usar uma cache conjunta com a gestão de Memória Virtual → Unified Page Cache
  - Gestão conjunta e dinâmica

Recordando: Periféricos tipo bloco

- Tipicamente discos rígidos – latência elevada, taxa de transferência melhor se sequencial
- Muitas vezes repetem-se acessos ao mesmo bloco
  - Por exemplo, o inode e o bloco da directoria raiz
- Guardar os blocos em memória reduz o número de acessos ao disco
- A cache/buffer de blocos (*block buffer cache*) tem duas funções:
  - Reservatório (pool) de **buffers** para E/S em curso
  - **cache** para operações de E/S já terminadas
- Atrasar as escritas permite juntar pedidos
  - Transformar várias escritas numa e/ou de vários blocos
  - **Permite também gerir esta fila de pedidos para um melhor desempenho**

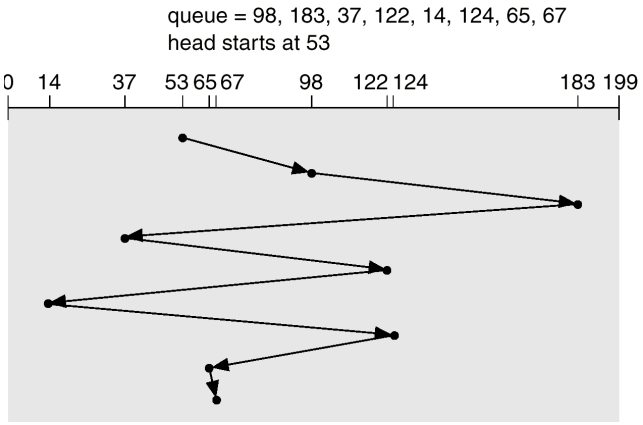
Escalonamento dos acessos ao disco

- Para cada disco há um certo número de pedidos de acesso pendentes
  - de um ou vários processos; sobre dados e meta-dados
- A reordenação da fila pode conduzir a tempos médios de acesso menores do que os conseguidos com FIFO/FCFS (*first come first served*)
- O tempo para atender cada pedido pode ser estimado (*seek time* e *rotation delay*)
- Exemplos de métricas para avaliar a qualidade dos algoritmos são:
  - Pedidos por unidade de tempo
  - Tempos de *Turnaround* e *Response*, ou taxas de transferência, para *workloads* pré-definidos
  - Número de pistas atravessadas pela cabeça (seek)

Escalonamento dos pedidos

- Exemplo:
- Considerando as pistas percorridas num conjunto fixo de pedidos
- Cenário:
  - Um disco com pistas de 0 a 199
  - Posição corrente da cabeça: pista 53
  - Sequência de pedidos para as pistas 98, 183, 37, 122, 14, 124, 65, 67

FCFS (ou FIFO)



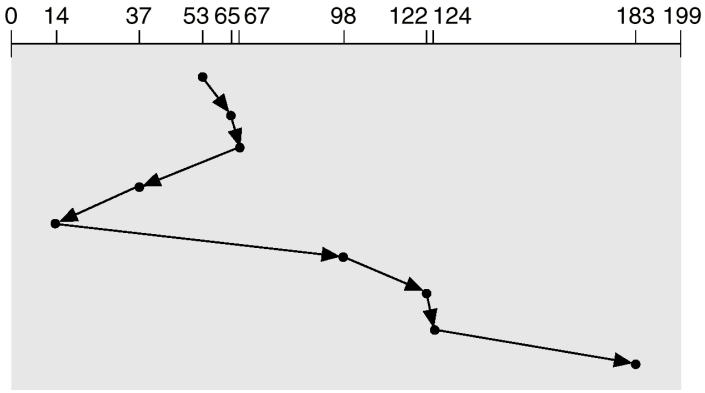
A cabeça movimenta-se sobre um total de 640 pistas

SSTF (shortest seektime first)

- Selecciona o pedido que corresponde à pista mais próxima da posição corrente da cabeça.
- SSTF pode causar esperas muito longas (ou *starvation*) a alguns pedidos.

SSTF (Cont.)

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53



No exemplo o total de movimentos da cabeça é 236 pistas

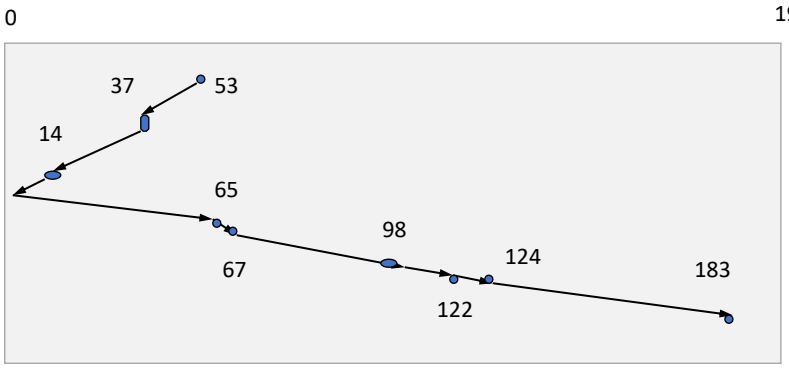
SCAN

- A cabeça começa movendo-se numa direção, executando pedidos até chegar ao extremo do disco; inverte o movimento da cabeça satisfazendo os pedidos pendentes.
  - atende sempre o pedido mais próximo no sentido em curso, alternado de sentido
- Também conhecido por **algoritmo do elevador**
- Impede *starvation*

SCAN (Cont.)

Fila = 98, 183, 37, 122, 14, 124, 65, 67

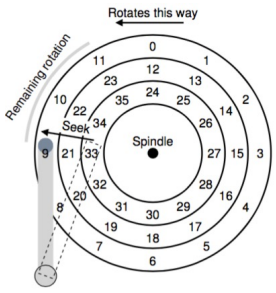
Cabeça = pista 53 (a descer)



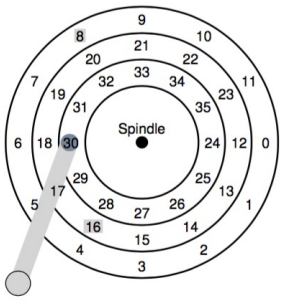
No exemplo, o total de movimentos da cabeça é de 208 cilindros.

Outras variantes

- C-SCAN – Circular SCAN
  - Sempre no mesmo sentido → evitar injustiça
- SPTF – Shortest Position Time First ou SATF - Shortest Access Time First
  - Não depende só da pista mas também da sua posição no disco



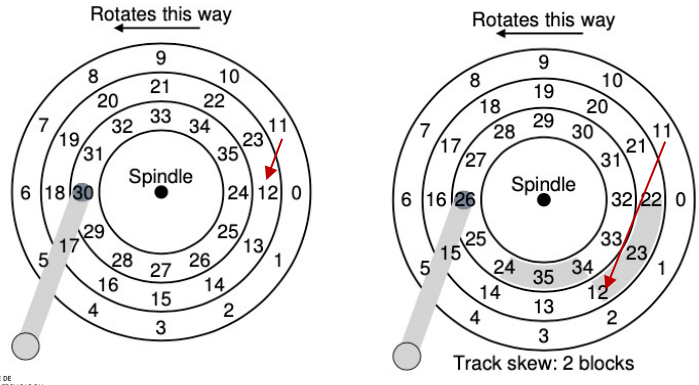
Latência depende do seek e da rotação



Qual está mais próximo do bloco 30? 16 ou 8?

Geometria do disco

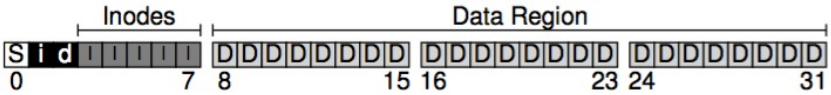
- Nem sempre a geometria é conhecida mas o acesso a sectores sequenciais é normalmente mais eficiente
- Renumerando para reduzir espera entre blocos consecutivos em pistas diferentes:



Conclusão

- SCAN, SPTF ou semelhante, tendo em conta a organização interna do disco
  - Prioridade à sequência dos blocos, tirando partido da reordenação feita pelo controlador interno do disco (este sabe a posição da cabeça, sabe onde estão os blocos, etc)
- Tirar partido de buffers/cache para reduzir número de acessos
- Não fazer o acesso imediatamente, em particular as escritas
  - SO fazer a fusão (merging) de pedidos contíguos
  - Aumentar a fila de pedidos para melhorar o escalonamento

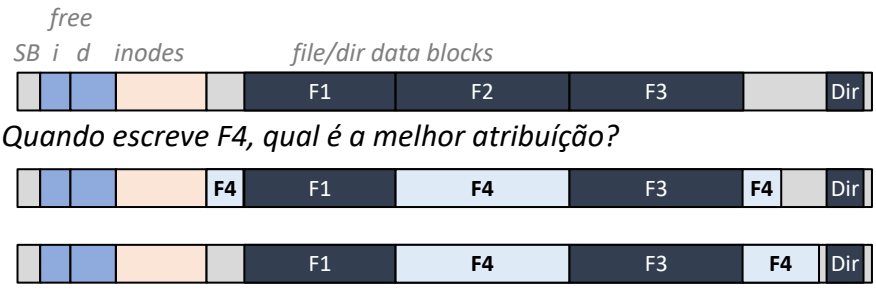
Afetação de blocos do disco



- Cada volume de disco inclui:
  - Boot block: se for caso disso, nos 1ºs sectores, código de arranque do SO
  - Super block: info sobre o SF (id do SF, sua dimensão, nº de inodes e de blocos de dados, dimensão dos blocos, *root inode*, etc.)
  - Mapa do uso de inodes e data: info sobre que inodes e blocos de dados estão em uso (e livres)
  - Inodes: info de todos os nós (ficheiros, diretorias, etc),
  - Blocos de Dados: os blocos efetivos com o conteúdo das diretorias e dos ficheiros

Afetação de blocos do disco

- A política de afetação dos blocos do disco aos ficheiros e diretorias influencia muito o desempenho da sua escrita e futuras leituras
- A fragmentação deve ser combatida
- A fragmentação agrava-se durante a utilização do disco

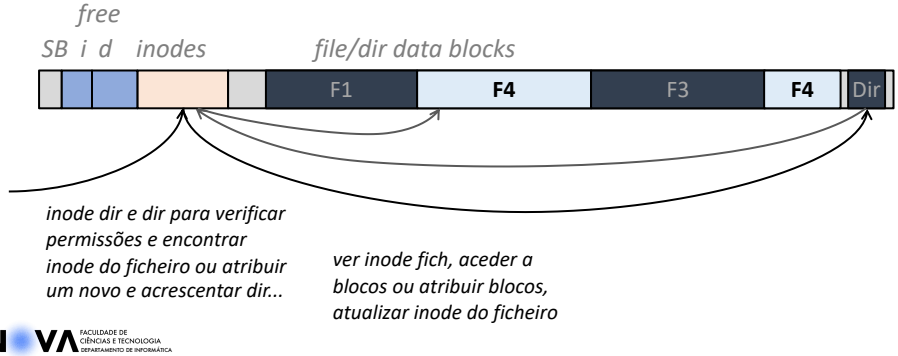


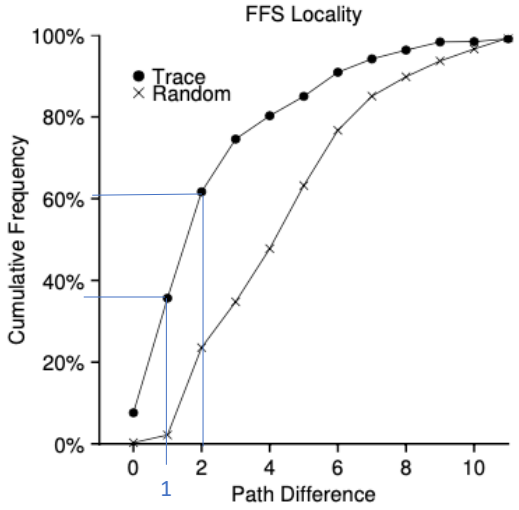
Criação, escrita e abertura de ficheiro

- Criação e abertura:
  - Percorre as diretorias e inodes para verificar permissões e encontrar o ficheiro → obter o seu inode
  - Para criar tem ainda de descobrir inode livre, acrescentar nome numa dir, atualizar bipmap, diretoria e inodes
- A cada escrita:
  - Se necessário, descobre bloco livre e afeta ao ficheiro, atualiza bloco, atualiza inode
  - Escolher os primeiros disponíveis de todo o disco?
    - Tende a usar mais o inicio do disco, não combate a fragmentação (agrava!)
- O senso comum aponta para localidade nos acessos com base nos nomes dos ficheiros/diretorias
  - Cada utilizador trabalha normalmente numa diretoria
  - Cada processo trabalha na sua working directory

Acessos a disco

- O acesso a um ficheiro (especialmente escrita) exige constantes leituras/escritas de dados, inode e bitmap de blocos livres
  - Estes estão espalhados pelo disco
  - Grave, especialmente para discos maiores ou mais ocupados (e fragmentados)

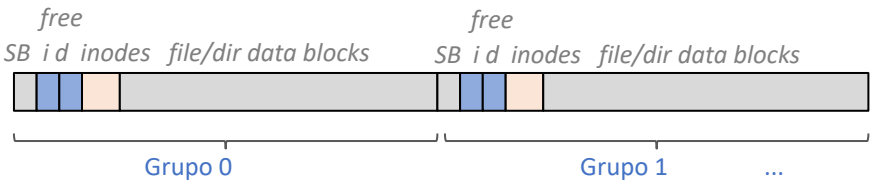




21

Afetação de blocos (Fast File System)

- Procurar manter juntos os blocos relacionados, quer sejam dados, diretorias ou inodes
  - Organizar o volume em grupos, distribuindo também os metadados (também ganha em backup do SB)



  - Criar diretoria: atribuir inode e dados do grupo com menos diretorias e mais inodes livres
  - Criar ficheiro: atribuir inode e dados do grupo da sua diretoria

exemplo

- **Criar diretorias e ficheiros** (c, d, e, f são ficheiros)
  /a/c, /a/d, /a/e, /b/f
- Antes:

Inodes *Dados*

/,a,c,d,e,b,f----------/,a,c,d,e,b,f,----------

- Com grupos:

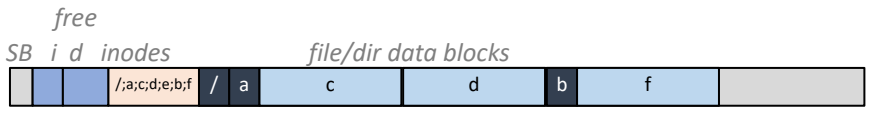
Inodes0 *Dados0* *Inodes1* *Dados1*

/,b,f,----------/,b,f,----------a,c,d,e,----------a,c,d,e,----------

FACULDADE DE CIÊNCIAS E TECNOLOGIA DEPARTAMENTO DE INFORMÁTICA

23

- P.e. A escrita de f exige constantes leituras/escritas no inode, no bipmap de dados, e de blocos de dados



- Com grupos essas operações são mais localizadas

FACULDADE DE CIÊNCIAS E TECNOLOGIA DEPARTAMENTO DE INFORMÁTICA

24

Caso de ficheiros grandes

- Ficheiros muito grandes tendem a desequilibrar os grupos (ou mesmo encher um grupo)
  - Impede outros da mesma diretoria de partilhar o grupo ou fragmenta-os
  - Acabará por ficar fragmentado se esgotar espaço do grupo
- Controlar, neste caso, a divisão do ficheiro por grupos
  - Colocar p.e. os blocos diretos no grupo do inode; o bloco de indiretos com os respetivos blocos noutro grupo; etc
    - O ficheiro é distribuído em grandes fatias por vários grupos