Discos magnéticos

Hardware do disco:

Mais comuns: discos rígidos

Característica principal:

Leitura e escritas rápidas

⇒ Ideais para memória secundária (paginação, SAs, etc)

Discos óticos:

CD-ROMs, CDs graváveis e DVDs

- Distribuição de programas
- Jogos
- Filmes

Discos magnéticos

Hardware do disco: organização

- Cilindros
- Trilhas (nº de cabeçotes determina o nº de trilhas por cilindro)
- Setores (8 a 32 nos discos flexíveis a centenas nos discos rígidos)

  - ✓ Discos mais antigos: eletrônica quase toda no controlador
  - ✓ Discos mais novos: eletrônica (microcontrolador) na própria unidade de disco:
    - » Ex: IDE (Integrated Drive Electronics), SATA

Discos magnéticos

Hardware do disco: organização

Discos mais novos: possibilidade de posicionamentos simultâneos

Enquanto espera por posicionamento (busca) em um disco, o controlador pode começar o posicionamento em outro disco

⇒ ***REDUÇÃO CONSIDERÁVEL NO TEMPO MÉDIO DE ACESSO***

Discos magnéticos

Discos mais novos:

Também podem ler ou escrever em um disco enquanto esperam o posicionamento em um ou mais discos

- Controlador de disco flexível não pode ler ou escrever em um ou mais discos ao mesmo tempo
- Controladores de disco rígidos podem operar paralelamente ***até o ponto de transferência dos dados para o buffer do controlador***

⇒ **Somente uma transferência por vez para a memória principal é possível**

SISTEMAS OPERACIONAIS MODERNOS
3ª EDIÇÃO

Discos magnéticos

Parâmetro	Unidade de disquete IBM PC 360 KB	Disco rígido Western Digital WD 18300	WD 3000
Número de cilindros	40	10601	36481
Trilhas por cilindro	2	12	255
Setores por trilha	9	281 (em média)	63
Setores por disco	720	35742000	586072368
Bytes por setor	512	512	512
Capacidade do disco	360 KB	18,3 GB	300 GB
Tempo de busca (cilindros adjacentes)	6 ms	0,8 ms	0,7 ms
Tempo de busca (em média)	77 ms	6,9 ms	4,2 ms
Tempo de rotação	200 ms	8,33 ms	6 ms
Tempo para parada/início do motor	250 ms	20 ms	
Tempo de transferência de um setor	22 ms	17 μs	1,4

Tabela 5.3 Parâmetros de disco para a unidade de disquete do IBM PC 360 KB e para o disco rígido do Western Digital WD 18300.

Diferentes velocidades de evolução

Discos magnéticos

HW do disco: organização

- Discos modernos são divididos em zonas e a qtde de setores pode variar de zona para zona (mais setores nas zonas externas)

- Geometria virtual, usada pelo ***driver (SW)*** pode ser diferente da geometria física

  ⇒ controlador é responsável por fazer o mapeamento no cilindro, cabeçote e setor reais

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

192 setores em ambos os casos

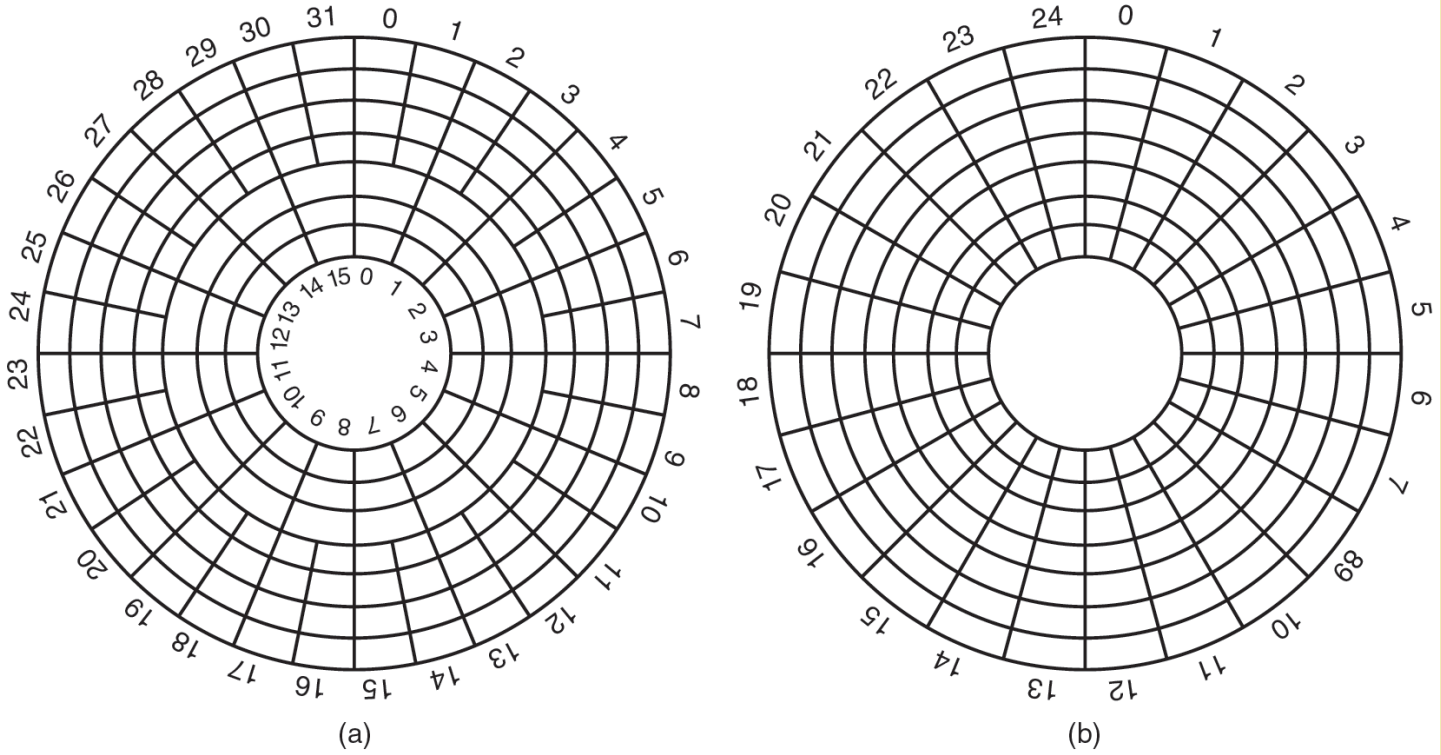


Figura 5.16 (a) Geometria física de um disco com duas zonas. (b) Uma possível geometria virtual para esse disco.

RAID

Desempenho das CPUs X desempenho dos discos

1. 1ª dobra a cada 18 meses, em média
2. desempenho dos discos: 10 vezes melhor desde 1970 (50 a 100 ms p/ < 10)

⇒ diferença entre desempenhos de CPU e disco aumenta a cada ano!!

Ideia: usar pararelismo também nos discos (Patterson, 1988)

→ RAID (Redundant Array of Independent Disks)

- ✓ caixa cheia de discos
- ✓ controlador RAID
- ✓ parece um SLED (Single Large Expensive Disk) p/ SO

RAID

RAID (Redundant Array of Independent Disks)

- ✓ Sistemas mais sofisticados: controlador RAID + 7 ou 15 discos SCSI, que possuem bom desempenho
- ✓ Dados são distribuídos pelos diversos discos
- ✓ Para o SO, o RAID parece um grande disco único com melhor desempenho e maior confiabilidade

  ⇒ **não é necessária nenhuma mudança no SW**

RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

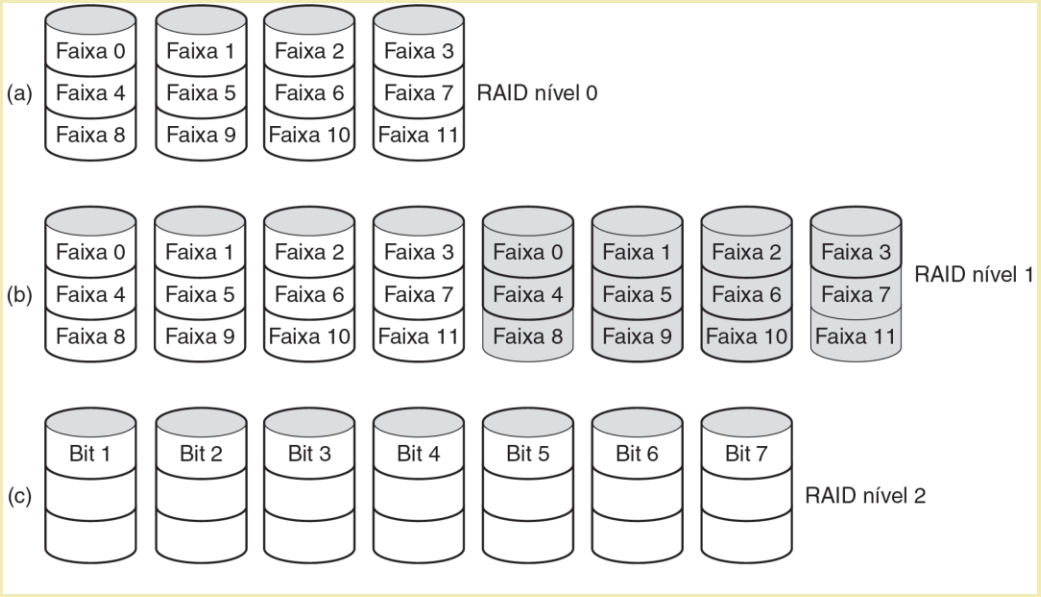
RAID nível 0:

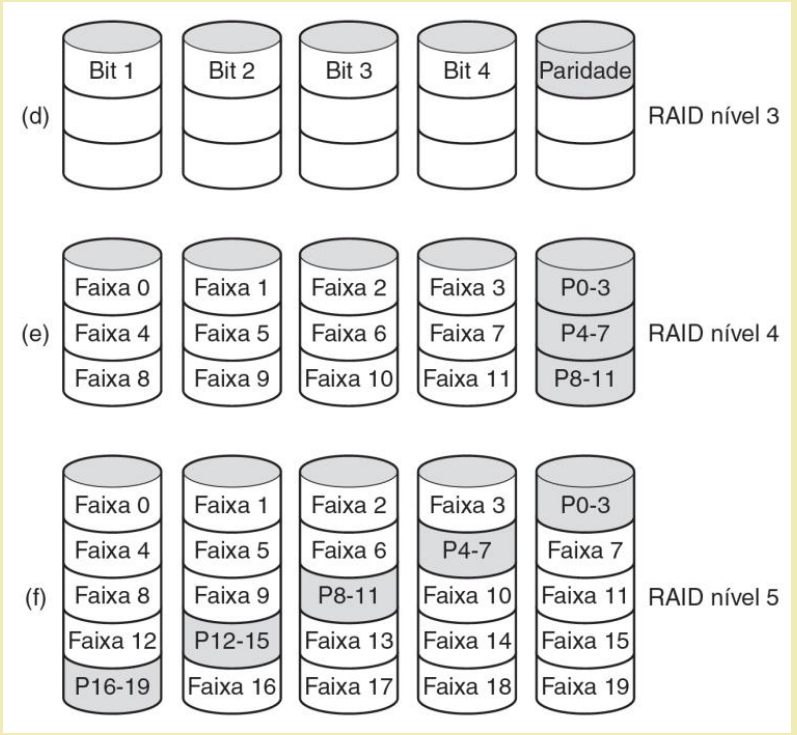
- cada faixa tem k setores;
- setores de 0 a k-1 na faixa 0, de k a 2k-1 na faixa 1, etc
- organização das faixas em alternância circular (*round-robin*)

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

RAID





RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

RAID nível 0: ***disco virtual é dividido em faixas de k setores***

Ex.: SW emite comando para gravar uma quantidade de blocos que ocupa 4 faixas → controlador RAID "quebra" comando do SW em 4 comandos separados, um por disco

⇒ E/S paralela transparente ao SW

- desempenho do RAID nível 0 é melhor quanto maior for a requisição
- uma requisição de bloco "muito grande" pode ser "quebrada" (pelo controlador) em várias requisições menores (transparente ao SW)

RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

RAID nível 0:

Desvantagens:

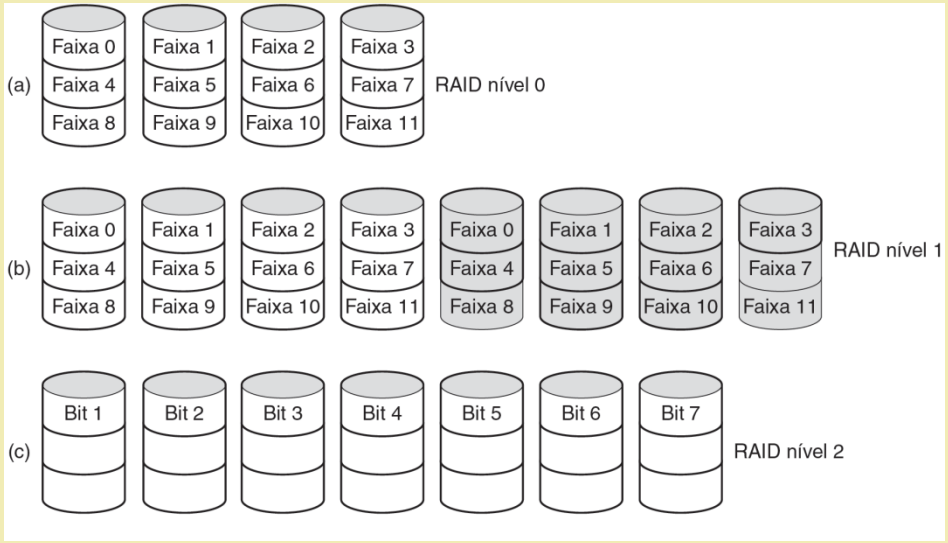
1. caso o SO requisite dados um setor de cada vez o desempenho cai, pois, neste caso, não há paralelismo

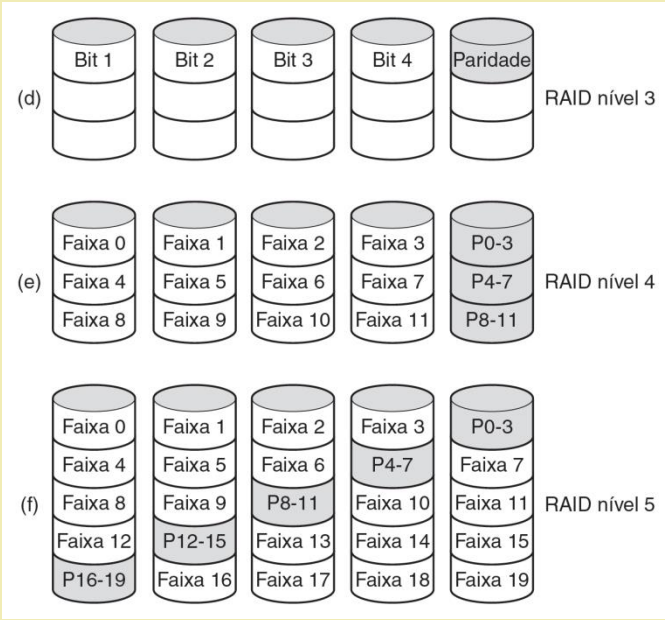
2. confiabilidade menor (4 discos com tempo médio de falha de 20 mil horas $\Rightarrow$ 1 falha a cada 5 mil horas)

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

RAID





RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

RAID nível 1:

- duplica todos os discos: quatro discos primários e quatro de backup
- escrita é realizada duas vezes
- leitura pode ser feita de qualquer dos 2 discos

⇒ desempenho da escrita não é melhor que em 1 cópia

⇒ desempenho da leitura pode ser até 2 vezes melhor

⇒ excelente tolerância a falhas e fácil recuperação

de um disco: usa o backup enquanto faz a instalação de um novo disco! Depois copia o backup para o novo!

RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

RAID nível 2:

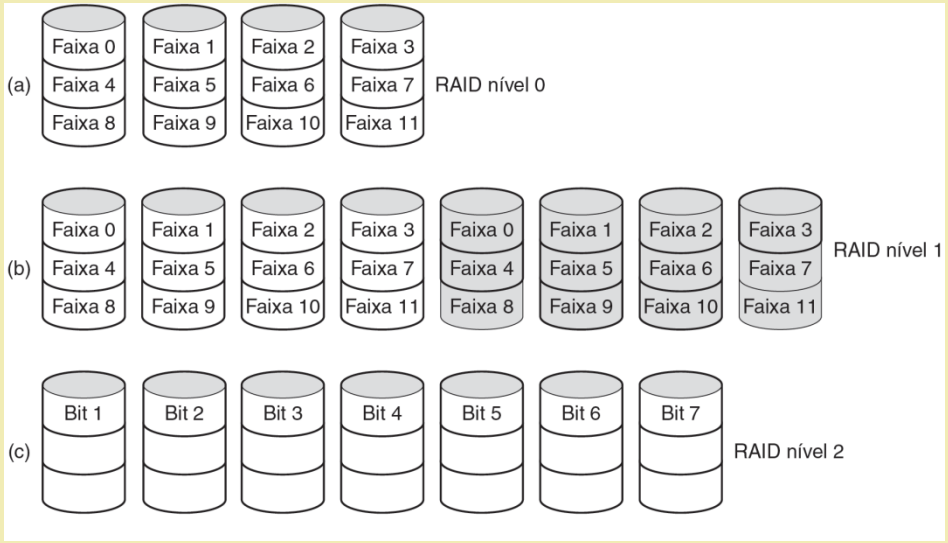
- ✓ escreve por palavra (ou bytes), dividindo a palavra entre os discos
- ✓ utiliza o código de Hamming para redundância (bits de paridade)

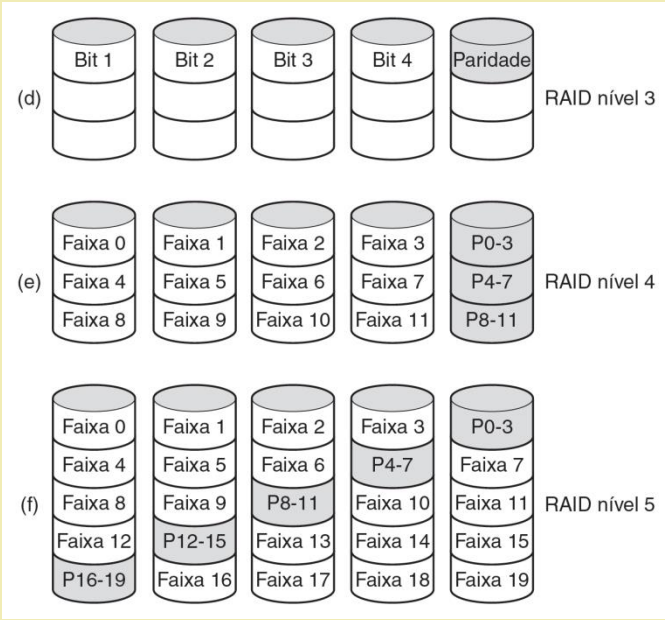
Ex. Computador CM-2 ⇒ palavras de 32 bits + 7 bits de paridade distribuídos em 39 discos

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

RAID





RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

RAID nível 2:

Vantagens:

1. no tempo de acesso a 1 setor ⇒ escreve em 32!
2. redundância: perda de um disco = perda de 1 bit

Desvantagens:

1. sincronização das rotações
2. sobrecarga causada pela codificação
3. controlador tem que fazer verificação de erro a cada bit recebido

RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

RAID nível 3:

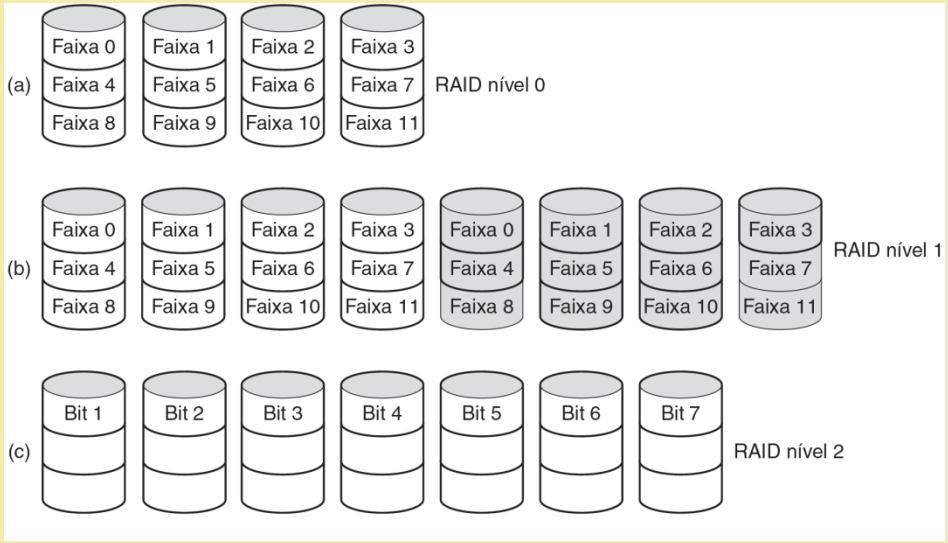
- ✓ versão simplificada do RAID nível 2
- ✓ um único bit de paridade é calculado e gravado num disco separado (disco de paridade)
- ✓ discos sincronizados, assim como no RAID nível 2
- ✓ não possibilita a correção de erros aleatórios
- ✓ correção do erro somente é possível quando existe a quebra de um disco, pois a posição do erro é conhecida

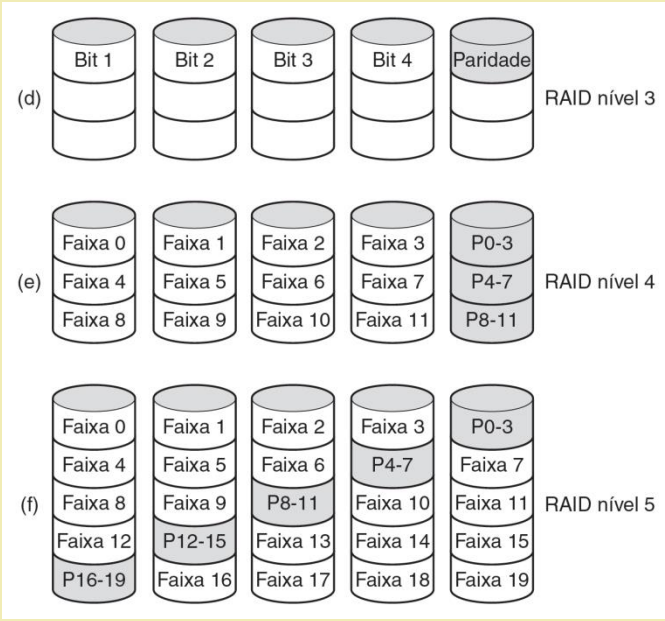
Obs: Raids 2 e 3: taxa de transf. alta mas num. de solicitações separadas por seg. não é melhor do que para um disco único

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

RAID





RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

RAID nível 4 :

- ✓ também trabalha com faixas (assim como o RAID nível 0), não necessitando da sincronização dos discos
- ✓ similar ao RAID nível 0, mas com um disco extra de paridade
- ✓ dados perdidos com a quebra de um disco podem ser recuperados a partir do disco de paridade, por meio da leitura de todos os outros discos.

Principal vantagem: protege contra a perda de um disco!

RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

RAID nível 4 :

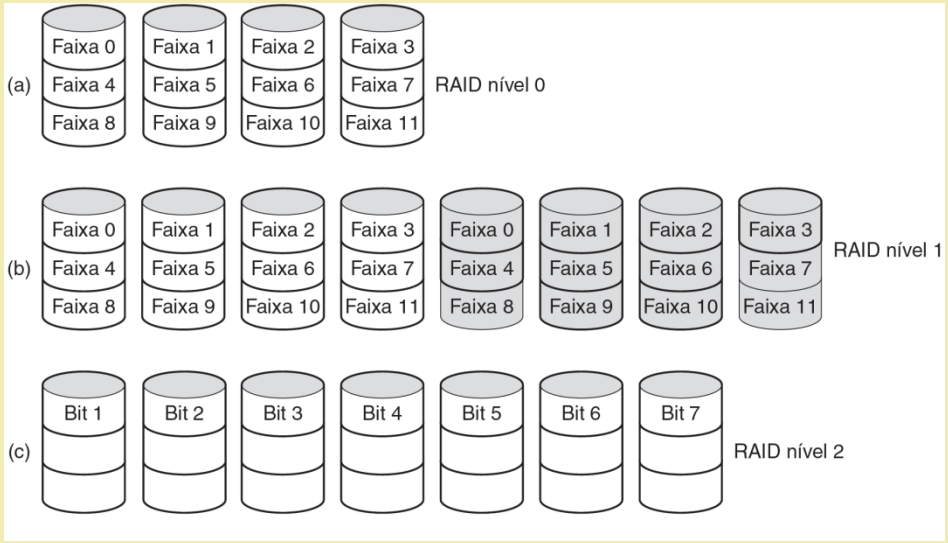
Desvantagem:

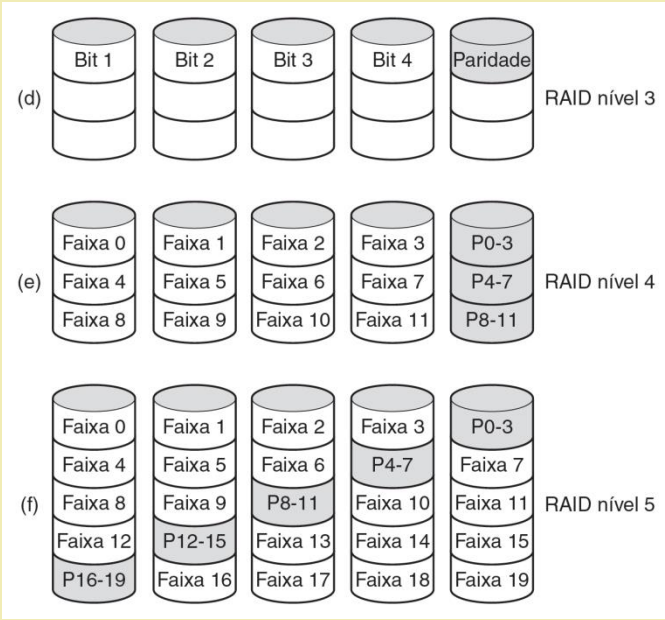
A alteração de apenas 1 setor exige a leitura de todos os discos para recalcular a nova paridade, além de nova escrita no disco de paridade ⇒ ***carga de trabalho pesada no disco de paridade!***

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

RAID





RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

RAID nível 5 :

- no RAID nível 4, o disco de paridade ***pode se tornar um gargalo!!***
- neste projeto os bits de paridade são distribuídos entre os discos
- diminui a carga para pequenas falhas (vantagem),
  mas na quebra de um disco a recuperação do mesmo é complexa
  - ⇒ perda de desempenho (desvantagem)
  - ⇒ erro em algum bloco no momento em que o disco está sendo reconstruído causaria problemas

RAID

RAID (Redundant Array of Independent Disks): níveis de 0 a 6

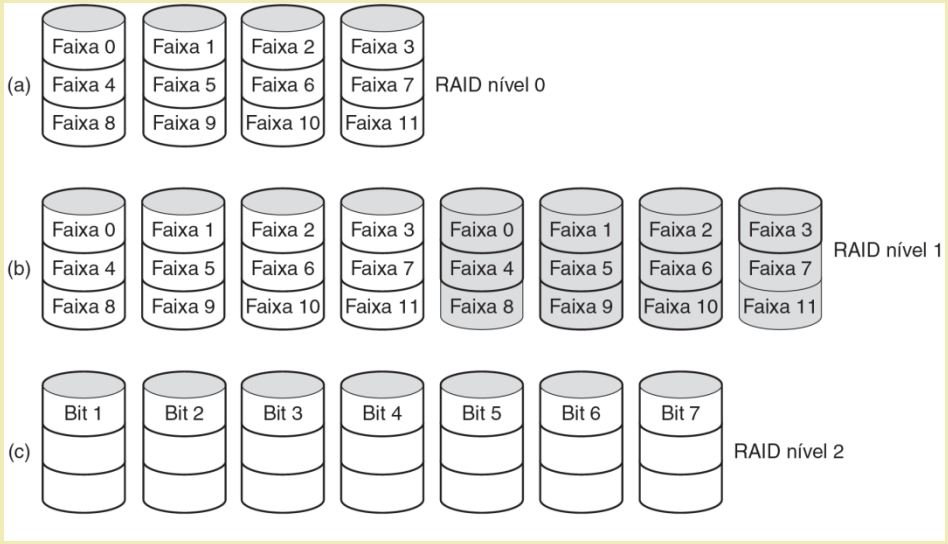
RAID nível 6 :

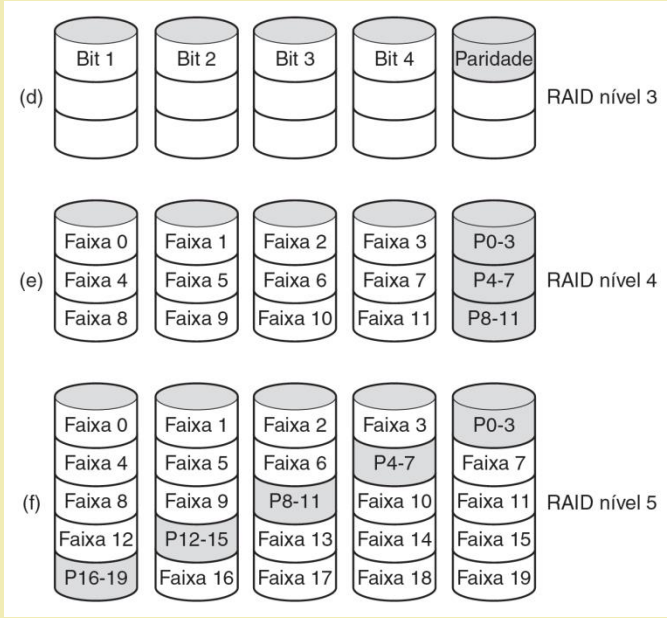
- similar ao RAID nível 5.
- adicionado um bloco de paridade extra em cada disco
- cada disco tem dois blocos de paridade (com a faixa extra em um disco diferente)
  - $\Rightarrow$ escritas um pouco mais caras
  - $\Rightarrow$ leituras não têm perda de desempenho, como no RAID nível 5

SISTEMAS OPERACIONAIS MODERNOS

3ª EDIÇÃO

RAID





Formatação de disco

Formatação de baixo nível

"consiste em criar uma série de trilhas concêntricas, cada uma com um certo número de setores, com pequenos intervalos entre eles"

SISTEMAS OPERACIONAIS MODERNOS
3ª EDIÇÃO

Formatação de disco

Formato de um setor

Figura 5.22 Um setor de disco.

Preâmbulo – padrão que permite ao hardware **reconhecer o início do setor:** também contém o nº do cilindro, nº do setor, etc

Dados – tamanho variável, determinado pelo programa de formatação

Usualmente: 512 B

ECC – código de correção de erros, usado para a recuperação de erros de leitura. Tamanho depende do controlador e do grau de confiabilidade desejado.

Usualmente: 16 B

Formatação de disco

Formatação de baixo nível

$\Rightarrow$ reduz a capacidade do disco!!

- tamanho do preâmbulo
- tamanho do intervalo entre os setores
- tamanho do ECC
- número de setores sobressalentes reservados

$\Rightarrow$ normalmente 20% menor do que a capacidade sem formatação!

Formatação de disco

Formatação afeta o desempenho do disco!

Ex: disco de 10.000 rpm, 300 setores por trilha, de 512 bytes cada um,

⇒ 6 ms para ler os 153.600 bytes de uma trilha ⇒ taxa = 24,4 MB/s - **taxa máxima**, independentemente da velocidade da interface ser maior!

Problema: se buffer só tem capacidade para um setor e o comando pediu a leitura de 2 setores

⇒ **Após primeira leitura, transferência para memória principal**

Mas enquanto a transferência é feita, setor a ser lido em seguida passa para pelo cabeçote

⇒ **Espera quase mais uma rotação completa!!!**

Formatação de disco

Solução 1

Entrelaçamento simples:

Dá um tempo ao controlador para que este faça a cópia do buffer para a memória principal - Fig. 5.24b

Solução 2

Entrelaçamento duplo:

Dá mais tempo, para o caso da transferência ser muito lenta - Fig. 5.24c

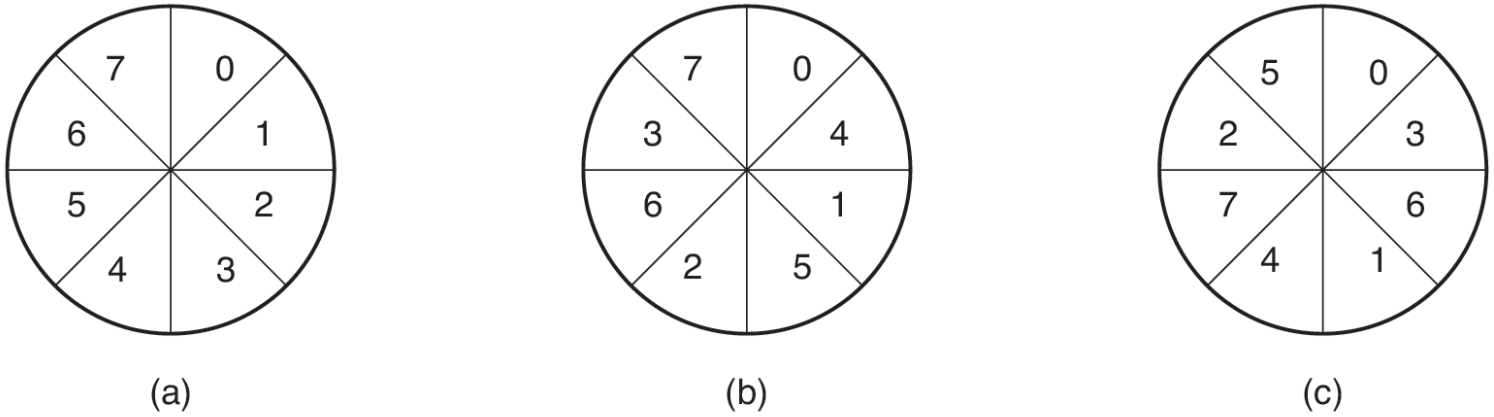


Figura 5.24 (a) Sem entrelaçamento. (b) Entrelaçamento simples. (c) Entrelaçamento duplo.

Formatação de disco

Partições do disco:

- cada partição é tratada como um disco separado
- viabilizam a coexistência de vários sistemas operacionais
- podem ser utilizadas como área de troca (swap) entre disco e memória
- setor 0 contém o MBR (***registro mestre de inicialização)***, com o código de boot e a tabela de partições (com setor inicial e tamanho de cada partição)
- uma das partições deve ser marcada como ativa na tabela de partições

Formatação de disco

Formatação de alto nível de cada partição:

– insere um bloco de inicialização

– define a estrutura de gerenciamento de armazenamento (*mapa de bits* ou *lista de blocos livres*)

– insere um diretório-raiz

– insere um sistema de arquivos vazio

– coloca um código na tabela de partições informando o tipo do sistema de arquivos usado na partição

Formatação de disco

Passos da inicialização:

- energia ligada: BIOS executa, carregando o MBR
- salta para o MBR e executa o programa de inicialização
- verifica qual a partição ativa
- lê o setor de inicialização daquela partição e o executa
- este setor contém um programa que carrega outro (maior, um ***carregador de inicialização***), que procura o sistema de arquivos para carregar o núcleo do SO
- SO é carregado na memória

Algoritmos de escalonamento de braço de disco

Fatores relacionados ao tempo de ler/escrever no disco

1. **Tempo de busca/posicionamento (o tempo para mover o braço para o cilindro correto)**
2. **Atraso na rotação (o tempo necessário para posicionar o setor correto sob o cabeçote)**
3. **Tempo de transferência real dos dados**

Algoritmos de escalonamento de braço de disco

Tempo de posicionamento é o preponderante na maioria dos discos!!

Objetivo dos algoritmos de escalonamento:

$\Rightarrow$ **Redução do tempo médio de posicionamento**

$\Rightarrow$ **melhora no desempenho do sistema**

Algoritmos de escalonamento de braço de disco

FCFS: requisições são atendidas na ordem em que chegam

$\Rightarrow$ **nada pode ser feito para melhorar o desempenho!!!**

Como melhorar?

Obs: Drivers contém uma tabela indexada pelo número do cilindro, onde cada entrada (cilindro) contém as requisições pendentes para o mesmo, encadeadas.

Algoritmos de escalonamento de braço de disco

Algoritmo SSF (*short seek first*) - posicionamento mais curto primeiro

próxima requisição a ser atendida é determinada pela requisição cujo cilindro estiver mais próximo do cilindro da requisição atual, ou seja, mais próxima da posição atual do cabeçote de leitura/gravação

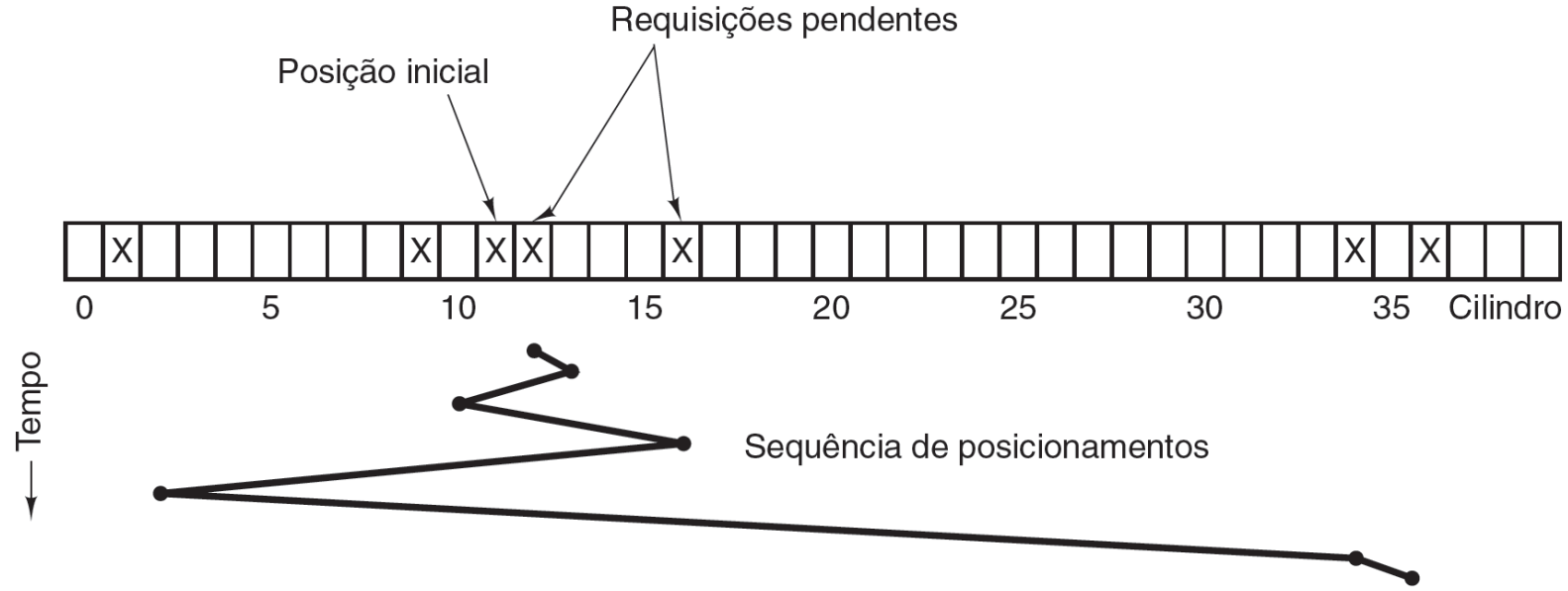


Figura 5.25 Algoritmo de escalonamento 'posicionamento mais curto primeiro' (SSF).

Ex: enquanto o posiciona para cilindro 11, chegam requisições 1, 36, 16, 34, 9, 12

FCFS – 111 cilindros movimentados

SSF (Shortest Seek First) – 61 "cilindros" –

Problema: com o disco totalmente carregado (muitas solicitações), braço tenderá a permanecer no meio, **prejudicando os cilindros das extremidades!!!**

SISTEMAS OPERACIONAIS MODERNOS
3ª EDIÇÃO

Algoritmo do elevador:

- ✓ SW mantém um bit da direção - Sobe/Desce
- ✓ Quando requisição é concluída, driver do disco verifica direção atual e caminha naquela direção até atender a todas as requisições
- ✓ Quando não existe mais requisições naquela direção, inverte o bit e caminha na outra direção

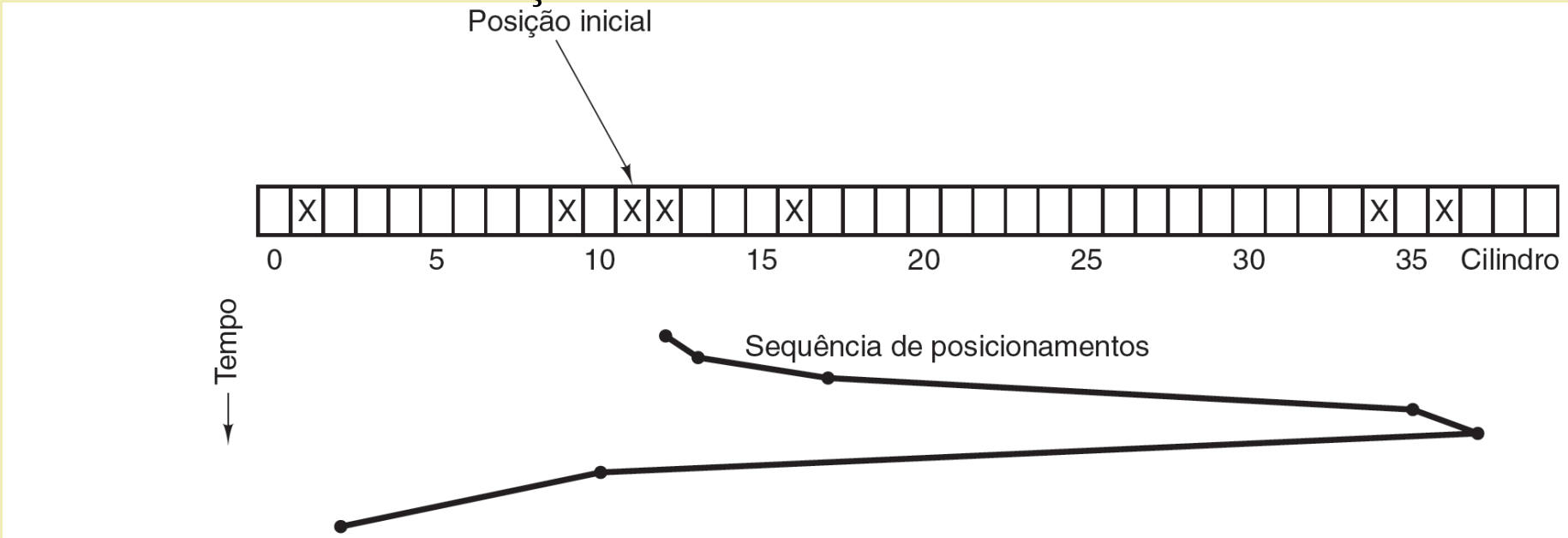


Figura 5.26 O algoritmo do elevador para escalonamento de solicitações do disco.

Algoritmo do elevador aplicado às requisições da Fig.: 60 cilindros!

Característica: valor máx. p/ distância é fixo: duas vezes o nº de cilindros!

Variação: **varrer os cilindros somente em uma direção!**

Observações:

1. Alguns controladores permitem ao SW saber o número do setor atual. Se duas requisições pertencem à mesma trilha, dá preferência;
2. Requisições para trilhas diferentes no mesmo cilindro não têm penalidade de tempo (escolha do cabeçote não envolve movimentação do braço)
3. Pode-se usar ***cache*** para evitar futuros reposicionamentos do braço. Cache conterá blocos que não foram de fato requisitados, ao contrário do que acontece com a cache do SO

Tratamento de erros

Aumento linear das densidades de bits

$\Rightarrow$ aumento nos defeitos de fabricação dos discos

$\Rightarrow$ setores defeituosos: ECC corrige poucos bits, mas não defeitos grandes

Sol. 1: tratar os blocos defeituosos via controlador

Sol. 2: tratar via SO

Sol. 1: lista de setores ruins é gravada na fábrica e substituídos por setores reservas

Tratamento de erros

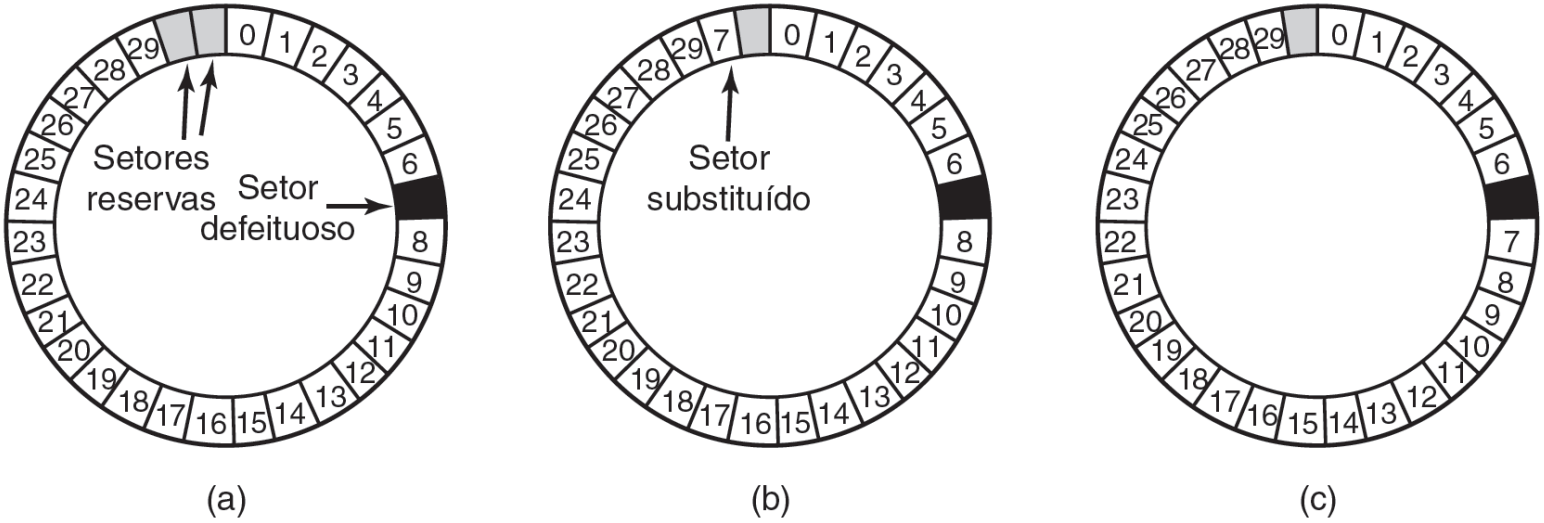


Figura 5.27 (a) Uma trilha de disco com setor defeituoso. (b) Substituição do setor defeituoso por um setor reserva. (c) Deslocamento de todos os setores para pular o setor defeituoso.

Tratamento de erros via controlador

Duas maneiras de fazer a substituição: 5.27b e 5.27c

- Em ambos os casos o controlador tem que saber qual setor é qual

- Dependendo da organização interna dos preâmbulos, segundo método é mais trabalhoso (reescrever 23 preâmbulos)
  ⇒ mas fornece desempenho melhor, pois uma trilha pode ser lida em uma única rotação

Tratamento de erros via controlador

Erros também surgem durante a operação

Tentativas de recuperação:

- Ler novamente o setor (no caso de poeira, apenas)
- Para erros repetitivos: trocar por um setor reserva antes da danificação completa ⇒ nenhum dado é perdido

Obs: Neste caso, primeira solução (5.27b) é usada, porque os outros setores podem conter dados

Tratamento de erros via SO

- SO deve ter certeza que setores defeituosos não estão em nenhum arquivo, nem na lista de blocos livres e nem no mapa de bits

  Solução: criar um arquivo secreto consistindo de setores defeituosos

- Arquivo não é conhecido pelo sistema de arquivos (nem pelos usuários!)

Problemas: 1. Backup do sistema, arquivo por arquivo!

Sol.: SO deve escondê-lo do utilitário de backup!

2. Backup setor por setor! Não tem como o SO esconder!

Sol.: Utilitário deve rejeitar setor após 10 tentativas de leitura e prosseguir copiando os setores restantes!

Tratamento de erros

- Erros de posicionamento do braço:
  corrigidos pela maioria dos controladores, caso contrário, o driver deve corrigir (comando ***recalibrate***)

- Controlador possui um pino no chip para que o *driver* possa reiniciá-lo se ocorrer algum erro

- Problema da recalibragem: **Sistemas de tempo real.**
  Exemplo: vídeo gravado em um CD-ROM
  $\Rightarrow$ bits devem chegar a uma taxa uniforme e recalibragem (que insere intervalos dentro de um fluxo de bits) se torna inaceitável!

Solução: uso de dispositivos especiais (discos AV) que nunca recalibram

Armazenamento estável

Ideal: disco nunca falhar

Problema:

Dados armazenados em virtude de uma operação de escrita podem estar corrompidos (ex: queda do Sistema provocada por uma falta de luz durante a escrita).

Objetivo: manter o disco consistente!

Armazenamento estável

Definição:

ocorre um *armazenamento estável*, quando o disco escreve corretamente ou não escreve nada, deixando os dados existentes intactos

Característica:

escrita é correta ou incorreta. Erro na escrita pode ser detectado na leitura seguinte, verificando os valores dos campos ECC dos setores

Armazenamento estável

Algoritmo para armazenamento estável

Suposições:

1. Probabilidade de um dado aleatório ter o mesmo ECC de 16 bytes ($\cong 2^{-144}$) é pequena o suficiente para ser considerada nula;

2. A probabilidade de dois erros acontecerem, simultaneamente, no mesmo setor/bloco de dois discos diferentes é considerada nula;

3. A CPU pode falhar e, neste caso, a escrita é interrompida, causando dados incorretos em um setor e gerando um ECC incorreto, podendo ser detectado posteriormente

Armazenamento estável

Sob as condições 1, 2 e 3 o armazenamento estável é 100% confiável, ou seja, ou as escritas trabalham corretamente ou deixam os dados antigos no local

Funcionamento:

- Usa par de discos idênticos trabalhando juntos
- Na ausência de erros, os blocos correspondentes em ambos os discos são iguais

  ⇒ a partir de qualquer um que seja lido se obtém o mesmo resultado

Armazenamento estável

Funcionamento (cont):

- Para se alcançar este objetivo (discos idênticos), define-se 3 operaçãos:

1. Escritas estáveis.
2. Leituras estáveis.
3. Recuperação de falhas.

Armazenamento estável

Algoritmo para escritas estáveis:

Escrever um bloco na unidade 1 / ler o mesmo bloco

Enquanto escrita não estiver correta (comparação de ECCs)

- ✓ Escrever e ler novamente (até *n* vezes) ou até obter um sucesso
- ✓ Se tiver *n* falhas consecutivas, bloco é remanejado sobre um bloco reserva e repete-se operação até a mesma ser bem sucedida (para todos os blocos reservas possíveis)

Armazenamento estável

Algoritmo para escritas estáveis:

Escrever na unidade 2 / ler na unidade 2

até obter um sucesso (de forma idêntica ao que foi feito para a unidade 1)

Resultado (na ausência de falhas da CPU):

⇒ **bloco escrito e testado em ambas as unidades**

Armazenamento estável

Algoritmo para leituras estáveis:

Ler primeiro bloco da unidade 1 e verificar ECC

Se ECC incorreto, tentar ***n*** vezes

Se todas as ***n*** leituras gerarem falhas, bloco é lido da unidade 2

Obs: lembrar que a probabilidade do mesmo bloco apresentar problema em ambas as unidades é próxima de zero (suposição 2)

Armazenamento estável

Algoritmo para recuperação de falhas

Se sistema tem uma falha

Programa de recuperação "varre" os dois discos comparando os blocos correspondentes:

- ✓ Se um par de blocos está bom e ambos são iguais ⇒ nada é feito
- ✓ Se um dos blocos apresenta erro de ECC, ele é reescrito com o bloco bom correspondente da outra unidade
- ✓ Se um par de blocos é bom (ECC correto) mas possuem conteúdos diferentes, bloco da unidade 1 é escrito sobre o bloco da unidade 2

Armazenamento estável

Algoritmo funciona mesmo com falhas da CPU

- ✓ falha antes da de uma das duas cópias ser escrita

  ⇒ na recuperação, valor antigo continua nas duas unidades

- ✓ falha durante a escrita na unidade 1

  ⇒ programa de recuperação detecta erro e restaura o bloco da unidade 1 a partir da unidade 2 ⇒ valor antigo restaurado;

- ✓ falha após escrita da unidade 1 mas antes da escrita na unidade 2

  ⇒ sem necessidade de desfazer operação. Programa de recuperação copia o bloco da unidade 1 (novo valor) para unidade 2

Algoritmo funciona mesmo com falhas da CPU

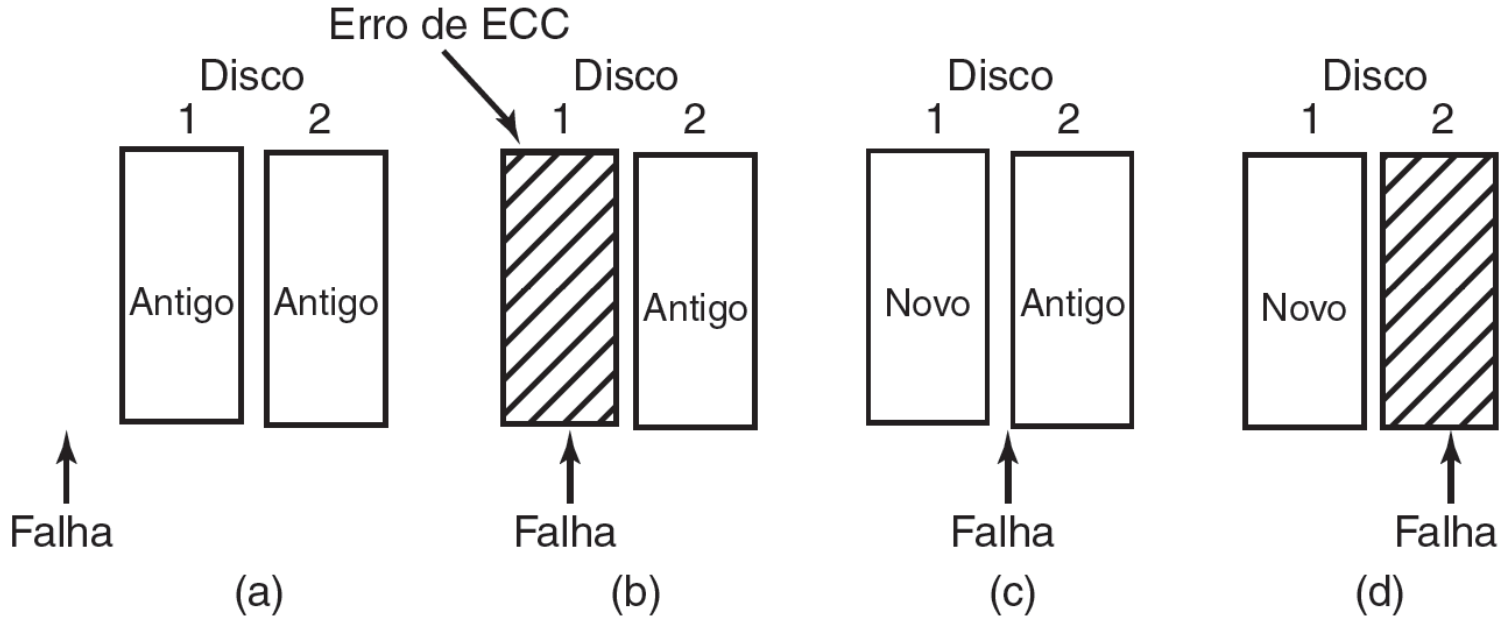


Figura 5.28 Análise da influência das falhas sobre as escritas estáveis.

Armazenamento estável

Algoritmo funciona mesmo com falhas da CPU (cont.)

- ✓ falha durante a escrita na unidade 2

  ⇒ programa de recuperação detecta erro e restaura o bloco da unidade 2 a partir da unidade 1 (ambas ficam com o novo valor);

- ✓ falha após a escrita na unidade 2

  ⇒ programa de recuperação vê que os blocos são iguais, nada a fazer

Armazenamento estável - otimização

Comparar bloco a bloco após uma falha é custoso!!!

- **Pode-se gravar uma cópia do bloco a ser armazenado no disco numa RAM não-volátil (ou num bloco separado caso ela não exista)**
- **Após a escrita ser feita corretamente coloca-se o valor -1 na RAM**
- **Após uma falha, programa de recuperação confere para ver se havia uma escrita estável em andamento e qual bloco estava sendo escrito quando ocorreu a falha**

$\Rightarrow$ Cópias são conferidas quanto à exatidão