INE5607 – Organização e Arquitetura de Computadores

Hierarquia e Gerência de Memória

Aula 29: Memórias e dependabilidade

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br

Sumário

- Dependabilidade
- Detecção de 1 erro
- Correção de 1 erro
- SECDED
- RAID

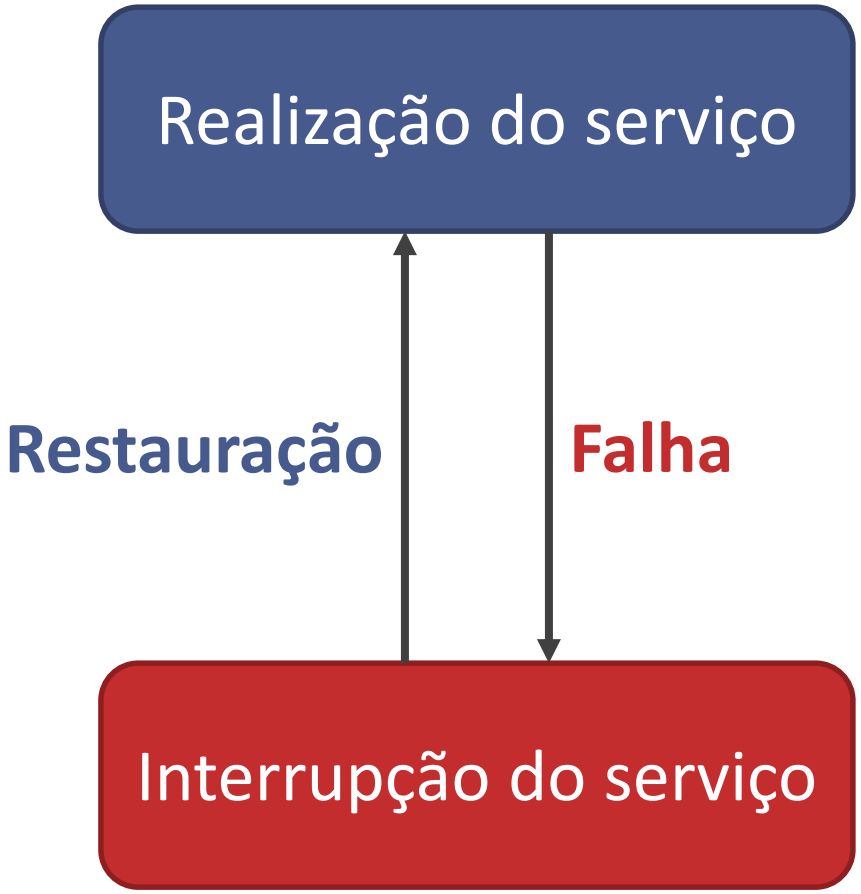
DEPENDABILIDADE

Dependabilidade

- **Definição informal**
  - Quanto podemos contar com um serviço cumprindo com suas especificações
- **Estados de um serviço**
  - **Realização do serviço**
    - Serviço trabalha como especificado
  - **Interrupção do serviço**
    - Serviço diverge da especificação

Dependabilidade

- **Falta**
  - Defeito em um componente
- **Erro**
  - Comportamento fora do esperado
- **Falha**
  - Mudança de estado do serviço



Dependabilidade

- **Exemplos de faltas que podem gerar falhas**
  - Bit em uma palavra em memória acaba alterado
  - Disco rígido morre
  - Estagiário desliga servidor para ligar cafeteira

Dependabilidade

- Definições relacionadas
  - **Confiabilidade**
    - *Reliability*
    - Medida da continuidade da realização do serviço
    - **Confiabilidade = MTTF**: *mean time to failure*
      - Tempo médio para falha
    - AFR: *annual failure rate*
      - Taxa de falha anual

Dependabilidade

- **Exemplo de confiabilidade**
  - Discos com MTTF de 1.000.000 horas
    - 1.000.000/(365*24) = 114 anos
  - Provedor de serviço com 50.000 servidores
    - Cada servidor com 2 discos
  - Quantos discos devem falhar por ano?
    - AFR = 365*24/1.000.000 = 0,00876 = 0,876%
    - 100.000 discos * 0,876% = **876 discos por ano**
      - **Mais do que dois por dia!**

Dependabilidade

- **Definições relacionadas**
  - **MTTR**: *mean time to repair*
    - Tempo médio para restaurar o sistema
    - Tempo de interrupção
  - **MTBF**: *mean time between failures*
    - Tempo médio entre falhas
    - **MTBF = MTTF + MTTR**

Dependabilidade

- **Definições relacionadas**
  - **Disponibilidade**
    - *Availability*
    - Medida de realização de serviço em relação a alternância entre os dois estados
    - **Disponibilidade = MTTF / (MTTF + MTTR)**

Dependabilidade

- **Disponibilidade em "noves"**
  - Um nove: 90% -> 36,5 dias de reparo/ano
  - Dois noves: 99% -> 3,65 dias de reparo/ano
  - Três noves: 99,9% -> 526 minutos/ano
  - Quatro noves: 99,99% -> 52,6 minutos/ano
  - Cinco noves: 99,999% -> 5,26 minutos/ano

Dependabilidade

- **Formas de melhorar disponibilidade**
  - Aumentar MTTF (reduzir ocorrências)
    - **Prevenção de faltas**
      - Por design; uso de componentes melhores
    - **Tolerância a faltas**
      - Uso de redundância para manter o serviço em realização
    - **Predição de faltas**
      - Troca de componentes antes que levem a falhas

Dependabilidade

- **Formas de melhorar disponibilidade**
  - Reduzir MTTR
    - Melhores ferramentas e processos
      - **Diagnóstico**
      - **Reparo**

DETECÇÃO DE 1 ERRO

Detecção de 1 erro

- **Richard Hamming**
  - Inventor de esquema de redundância para memórias
    - Razão de receber o **Turing Award** em 1968
  - Para detectar erros, é bom saber o quão "próximos" padrões de bits corretos são
  - **Distância de Hamming**
    - Mínimo número de bits que são diferentes entre duas sequências corretas de bits

Detecção de 1 erro

- **Exemplo**
  - 011011 e 001111
  - Distância de Hamming = 2
  - No caso de um erro que altere um dos bits, passamos de um código válido para um código inválido!
    - **Temos um código de detecção de erros em 1 bit**

Detecção de 1 erro

- **Paridade**
  - Mecanismo básico para detecção de erro em 1 bit
  - Adiciona-se **um bit** a palavra sendo protegida
  - Conta-se o número de bits em 1 na palavra
    - **Se impar, bit de paridade = 1**
    - **Se par, bit de paridade = 0**
  - Exemplo
    - 42 = 0b00101010 -> 0b001010101**1**
    - 6 = 0b00000110 -> 0b000001100

Detecção de 1 erro

- **Exercício**
  - Calcule a paridade do byte com valor 31
  - Suponha que o bit mais significativo é invertido. O erro é detectado?
  - O que acontece se os dois bit mais significativos forem invertidos?
    - 31 = 0b00011111 -> 0b000111111**1**

CORREÇÃO DE 1 ERRO

Correção de 1 erro

- **Correção de 1 bit errado**
  - Paridade não é o bastante
    - Qual bit está errado em 0b11110000**0**?
    - Só detecta, não corrige
  - **Distância de Hamming 3**
    - Caso um bit mude de valor, ele ainda estará mais próximo da palavra original do que de outras!

Correção de 1 erro

- **Código de Correção de Erros de Hamming**
  - **ECC**: *Error Correcting Code*
  - Mapeamento de dados de distância 3 fácil de entender
    - Bits de paridade no meio da palavra de dados
    - Permitem identificar o bit errado facilmente

Correção de 1 erro

- **Etapas**
  1. Numere os bits da palavra a partir de 1 da esquerda para direita
  2. Marque todas as posições que são potência de 2 para bits de paridade
  3. Todas as outras posições servem para dados
  4. A posição de um bit de paridade determina quais bits ele verifica
  5. Calcule a paridade de cada bit

Correção de 1 erro

- **Etapas**
  1. Numere os bits da palavra a partir de 1 da esquerda para direita

Posição dos bits	1	2	3	4	5	6	7	8

  2. Marque todas as posições que são potência de 2 para bits de paridade

Posição dos bits	1	2	3	4	5	6	7	8	9	10	11	12
Bits codificados	p1	p2		p4				p8				

  3. Todas as outras posições servem para dados

Posição dos bits	1	2	3	4	5	6	7	8	9	10	11	12
Bits codificados	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8

Correção de 1 erro

- **Etapas**
  4. A posição de um bit de paridade determina quais bits ele verifica
     - Bit 1 (0b0001): bits cuja posição tem o bit mais à direita em 1 (1,3,5,7,9,11,...)
     - Bit 2 (0b0010): bits cuja posição tem o segundo bit mais à direita em 1 (2,3,6,7,10,11,...)

Posição dos bits		1	2	3	4	5	6	7	8	9	10	11	12
Bits codificados		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Cobertura de bits de paridade	p1	X		X		X		X		X		X	
	p2		X	X			X	X			X	X	
	p4				X	X	X	X					X
	p8								X	X	X	X	X

Correção de 1 erro

- **Etapas**
  5. Calcule a paridade de cada bit
     - Exemplo: 7 (0b00000111)
       - p1 = 0 + 0 + 0 + 0 + 1 = 1
       - p2 = 0 + 0 + 0 + 1 + 1 = 0
       - p4 = 0 + 0 + 0 + 1 = 1
       - p8 = 0 + 1 + 1 + 1 = 1

Posição dos bits		1	2	3	4	5	6	7	8	9	10	11	12
Dado				0		0	0	0		0	1	1	1
Cobertura de bits de paridade	p1	X		X		X		X		X		X	
	p2		X	X			X	X			X	X	
	p4				X	X	X	X					X
	p8								X	X	X	X	X

Correção de 1 erro

- **Etapas**
  5. Calcule a paridade de cada bit
     - Exemplo: 7 (0b00000111)
       - p1 = 0 + 0 + 0 + 0 + 1 = 1
       - p2 = 0 + 0 + 0 + 1 + 1 = 0
       - p4 = 0 + 0 + 0 + 1 = 1
       - p8 = 0 + 1 + 1 + 1 = 1

Posição dos bits		1	2	3	4	5	6	7	8	9	10	11	12
Final		**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**
Cobertura de bits de paridade	p1	X		X		X		X		X		X	
	p2		X	X			X	X			X	X	
	p4				X	X	X	X					X
	p8								X	X	X	X	X

Correção de 1 erro

- **Como verificar e corrigir erros**
  - Calcula-se novamente as paridades
  - Caso algum bit seja diferente de 0
    - Organiza-se os bits de paridade calculados para gerar a posição errada : p8p4p2p1
  - Caso contrário, valor está livre de 1 bit errado

Correção de 1 erro

- **Exemplo**

Posição dos bits		1	2	3	4	5	6	7	8	9	10	11	12
Original		1	0	0	1	0	0	0	1	0	1	1	1
Errado		1	0	0	1	0	0	0	1	1	1	1	1
Cobertura de bits de paridade	p1	X		X		X		X		X		X	
	p2		X	X			X	X			X	X	
	p4				X	X	X	X					X
	p8								X	X	X	X	X

p1 = 1 + 0 + 0 + 0 + 1 + 1 = 1 (algo de errado)

p2 = 0 + 0 + 0 + 0 + 1 + 1 = 0 (OK)

p4 = 1 + 0 + 0 + 0 + 1 = 0 (OK)

p8 = 1 + 1 + 1 + 1 + 1 = 1 (algo de errado)

p8p4p2p1 -> 0b1001 -> posição 9 é o problema!

SECDED

SECDED

- **SECDED**
  - *Single Error Correction, Double Error Detection*
  - Detecta 2 bits, corrige 1
  - **Distância de Hamming 4**
    - **Hamming ECC + bit de paridade para a palavra toda**

SECDED

- **SECDED**
  - Possíveis resultados
    - ECC = 0 e paridade = 0 -> tudo OK
    - ECC ≠ 0 e paridade ≠ 0 -> um erro corrigível
    - ECC = 0 e paridade ≠ 0 -> bit de paridade errado
    - ECC ≠ 0 e paridade = 0 -> dois erros detectados

SECDED

- **Exemplo**

pt = 1 + 1 + 1 + 1 + 1 + 1 = 0

Posição dos bits		0	1	2	3	4	5	6	7	8	9	10	11	12
Original		**0**	**1**	**0**	0	**1**	0	0	0	**1**	0	1	1	1
Cobertura de bits de paridade	p1		X		X		X		X		X		X	
	p2			X	X			X	X			X	X	
	p4					X	X	X	X					X
	p8									X	X	X	X	X
	pt	X	X	X	X	X	X	X	X	X	X	X	X	X

SECDED

- **Exemplo**

Posição dos bits		0	1	2	3	4	5	6	7	8	9	10	11	12
Original		0	1	0	0	1	0	0	0	1	0	1	1	1
Dois erros		0	1	0	0	1	0	0	0	1	1	0	1	1
Cobertura de bits de paridade	p1		X		X		X		X		X		X	
	p2			X	X			X	X			X	X	
	p4					X	X	X	X					X
	p8									X	X	X	X	X
	pt	X	X	X	X	X	X	X	X	X	X	X	X	X

p1 = 1, p2 = 1, p4 = 0, p8 = 0 -> 0b0011

pt = 0

ECC detecta erro + paridade não detecta -> dois erros no código

RAID

RAID

- ***Redundant Array of Inexpensive (Independent) Disks***
  - Arranjo redundante de discos baratos (independentes)
  - Mostra um disco lógico com vários discos físicos
    - Paralelismo aumenta desempenho
    - Discos adicionais proveem armazenamento redundante
    - **Tolerância a faltas no armazenamento**

RAID

- **Nomenclatura usual**
  - **RAID X**, onde X é um número
  - Números identificam o esquema de redundância
- **RAID 0**
  - Sem redundância
    - Apenas fatia dados sobre múltiplos discos
    - ***Striping***
  - Aumenta desempenho

RAID

- **RAID 1: Espelhamento**
  - **N + N discos**, replica dados
    - Escreve no disco de dados e no disco espelho
    - Em caso de falha, lê dados do disco espelho
- **RAID 2: ECC**
  - **N + E discos** (exemplo: 10 + 4)
    - Divide dados em nível de bit entre N discos
    - Gera código de correção de erros de E bits
    - Muito complexo, não usado na prática

RAID

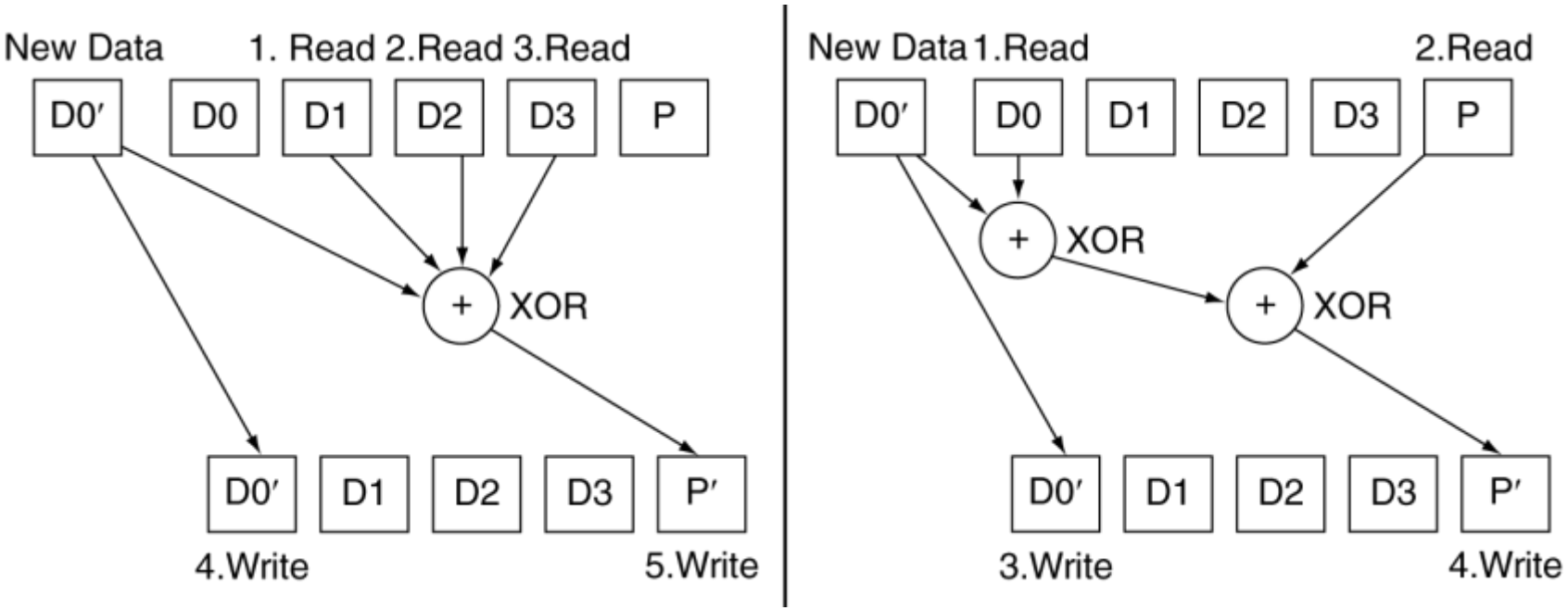
- **RAID 3: Paridade em nível de bit**
  - **N + 1** discos
    - Dados fatiados entre N discos em nível de **byte**
    - Disco redundante armazena paridade
    - **Leitura: Acessa todos os discos**
    - **Escrita: Gera nova paridade e atualiza todos os discos**
    - Em caso de falha: Usa paridade para recuperar dados
    - Não muito usado

RAID

- **RAID 4: Paridade em nível de bloco**
  - **N + 1** discos
    - Dados fatiados entre N discos em nível de **bloco**
    - Disco redundante armazena paridade para um grupo de blocos
    - **Leitura: Acessa apenas o disco com o bloco**
    - **Escrita: Gera nova paridade e atualiza o disco do bloco e o disco de paridade**
    - Em caso de falha: Usa paridade para recuperar dados
    - Não muito usado

RAID

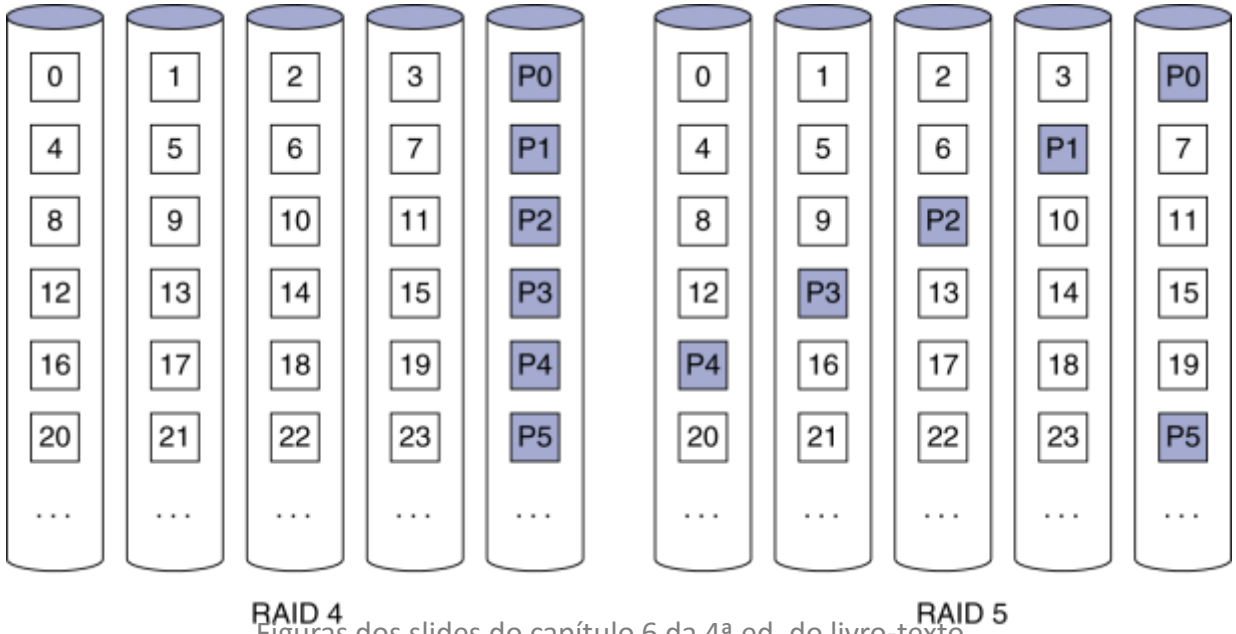
- **RAID 3 x RAID 4**
  - **RAID 4 otimiza escritas pequenas**
  - Porém ambos estressam o disco de paridade



Figuras dos slides do capítulo 6 da 4ª ed. do livro-texto.

RAID

- **RAID 5: Paridade em nível de bloco distribuída**
  - **N + 1** discos
    - Como RAID 4, mas **distribui os blocos de paridade**
    - Muito usado



RAID

- **RAID 6: Redundância P + Q**
  - **N + 2** discos
    - Como RAID 5, mas com **dois blocos de paridade**
    - Maior tolerância a faltas através de mais redundância
    - Algoritmos mais elaborados para cálculo de paridade

CONSIDERAÇÕES FINAIS

Considerações finais

- **Dependabilidade, confiabilidade e disponibilidade**
- **Hamming**
  - Distância de Hamming
  - Detecção de erro em 1 bit
  - Correção de erro em 1 bit
  - Correção de erro em 1 bit e detecção de erro em 2 bits
- **RAID**

INE5607 – Organização e Arquitetura de Computadores

Hierarquia e Gerência de Memória

Aula 29: Memórias e dependabilidade

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br