

FAULT TOLERANT SYSTEMS

<http://www.ecs.umass.edu/ece/koren/FaultTolerantSystems>

Part 2 - Algorithm-based Fault Tolerance

Chapter 3 - Information Redundancy

Part. 10.1

Copyright 2007 Koren & Krishna, Morgan-Kaufman

Algorithm-based Fault Tolerance (ABFT)

◆ Data redundancy at the application level

- * Higher efficiency when applied to large data arrays
- * **Examples:** matrix-based and signal processing applications
- * Given $n \times m$ matrix A define the column checksum matrix

$$A_C = \begin{bmatrix} A \\ eA \end{bmatrix} \quad \text{where } e = [111 \dots 1]$$

- * Row check matrix $A_R = [A \quad Af]$ where $f = [111 \dots 1]^T$
- * $(n+1) \times (m+1)$ full checksum matrix

$$A_F = \begin{bmatrix} A & Af \\ eA & eAf \end{bmatrix}$$

- * Column and row checksum matrices can detect a single fault
- * Full checksum matrix can locate a fault - if checksum accurate the fault can be corrected

Part. 10.2

Copyright 2007 Koren & Krishna, Morgan-Kaufman

ABFT for Matrix Operations

- ◆ Matrix addition $A+B=C$ can be replaced by

$$A_C + B_C = C_C \quad \text{or} \quad A_R + B_R = C_R \quad \text{or} \quad A_F + B_F = C_F$$

- ◆ Matrix multiplication $A \cdot B=C$

$$A B_R = C_R \quad \text{or} \quad A_C B = C_C \quad \text{or} \quad A_C B_R = C_F$$

- ◆ Faults can be located and corrected by adding weighted checksum row(s) or column(s)

$$A_C = \begin{bmatrix} A \\ eA \\ e_w A \end{bmatrix} \quad \text{where } e_w = [1, 2 \dots 2^{n-1}]$$

$$A_R = [A \quad Af \quad Af_w] \quad \text{where } f_w = [1, 2 \dots 2^{m-1}]^T$$

$$A_F = \begin{bmatrix} A & Af & Af_w \\ eA & eAf & eAf_w \\ e_w A & e_w Af & e_w Af_w \end{bmatrix}$$

Part. 10.3

Copyright 2007 Koren & Krishna, Morgan-Kaufman

Weighted Checksum Code (WCC)

- ◆ Example for single error correction:

$$A_C = \begin{bmatrix} A \\ eA \\ e_w A \end{bmatrix}$$

- * Suppose an error detected in column j
- * WCS1/WCS2 unweighted/weighted checksum $eA/e_w A$ for column j
- * Calculate error syndromes:

$$S_1 = \sum_{i=1}^n a_{i,j} - \text{WCS1} \quad S_2 = \sum_{i=1}^n 2^{i-1} a_{i,j} - \text{WCS2}$$

- * If only one syndrome is nonzero - the checksum is wrong
- * If both are nonzero $S_2/S_1 = 2^{k-1}$ implying that $a_{k,j}$ is in error
 $a'_{k,j} = a_{k,j} - S_1$

- * Extra rows (columns) can be added with

$$e_{w_d} = [1^{d-1} 2^{d-1} \dots (2^{n-1})^{d-1}] \quad f_{w_d} = [1^{d-1} 2^{d-1} \dots (2^{m-1})^{d-1}]^T$$

- * The weighted checksum with v rows - Hamming distance $v+1$: detecting v or correcting $\lfloor v/2 \rfloor$ errors

Part. 10.4

Copyright 2007 Koren & Krishna, Morgan-Kaufman

Checksum Overflow

- ◆ For large n and m checksums can overflow
 - * Single-precision l -bit checksum - result calculated mod- 2^l
 - * A single error $< 2^l$ and will be detected
 - * Weighted checksum would need more bits to avoid overflow
 - * Instead of $e_w = [1 \ 2 \ \dots \ 2^{n-1}]$ use $e_w = [1 \ 2 \ \dots \ n]$
 - * If both syndromes for column j are nonzero and $S_2/S_1 = k$
$$a'_{k,j} = a_{k,j} - S_1$$
 - * Round-off errors in floating-point can result in nonzero syndromes
 - * Must select d and only syndrome $> d$ will indicate an error
 - * Value of d : probability of fault detection vs. false alarms
 - * To simplify selection of d - partition into submatrices with separate checksums