Self-Repairing ECC Software

Center for Reliable Computing

Self-Repairing ECC Software
by
Philip P. Shirvani
RATS
January 10, 2000

Outline
- Motivation
- Software Duplication
- Cross-Checking ECC Programs
- Implementation in ARGOS
- Summary

Motivation
- COTS Board in ARGOS
  - Transient errors (SEUs) in memory
  - No hardware ECC
- Program Corruption
  - Operating system
  - Applications
- Repair Mechanism
  - Software ECC
    - One of the applications
    - Self-repair

Software-Implemented ECC [1][2]
- Code Segments
  - Fixed content after link stage
  - Generate ECC bits on the board
  - Scrub periodically
- Data Segments
  - Read-only data
  - Random reads and writes
    - Intercept store instructions
    - Inefficient in software
  - Through API

ECC Program Requirements
- Background Task
  - Transparent to other programs
- Fast and Small Program
  - Redundancy
  - Program complexity
- Prevent ECC Miscorrections
  - Due to bit-flips in ECC program
- Self-Repair
  - Increase availability of ECC protection

Software Duplication
- Run Two Copies and Compare Outputs
  - Miscompare indicates error
    - e.g., EDDI [3]
- Which One is Correct?
  - Self-check
    - (single error assumption)
- How to Repair the Faulty One?
  - ECC scrub
Self-Repairing ECC Software

Redundancy in ECC Software
- Self-Repair
  - Corrupt ECC program
  - Cannot repair itself
  - At least two copies
- Duplication
  - Task-level or EDDI
  - Detection but no correction
- Cross-Checking Pairs
  - Each copy scrubs the other one

Operations
- main_job
  - Scrubbing the blocks under protection
  - Mostly code segments of other programs
- scrub_ECCx
  - Scrubbing the other copy of ECC
- scrub_data
  - Scrubbing own data structure
  - Info on location of protected blocks
- self_check

Error Probabilities
- Observed SEUs (so far)
  - Mostly bit-flips in memory
  - ~10 bits/Mbyte-day = 10^{-18} per bit-cycle
- Periodic Scrubbing
  - Every 5 minutes; 2 sec. run-time
- Program Size < 50KB (two copies)
- Assuming Single Error
- “Scrub Before Use”
  - Very low chance of bit-flip during run-time

Implementation in ARGOS
- Periodic Scrubbing

Design Framework
- Multitasking
  - Separate, high priority task for two ECCs
- Synchronization
  - Timer for periodic scrubbing
    - ECC1 wake-up
  - Message-passing
    - ECC2 wake-up
  - Semaphore
    - “Done” signal to ECC1
Self-Check

- Run ECC on a Block of All 0’s
  - Check encoding routine (all-0 check bits)
  - Check error detection routine
  - Inject a single error
  - Check error correction routine
- If Error
  - Invoke other ECC program for repair
- Coverage Problem
  - Scrubbed before untested parts are used

Hang-up Errors

- Detection
  - Time-out in message-passing or semaphores
- Check if ECC task is suspended
- Recovery
  - Scrub errors
  - Kill and restart task
    - Data structures preserved
  - If Unsuccessful
    - Reload program

Other Self-Repairing Schemes for ECC

- Duplication
  - EDDI for each copy
    - Provides self-checking
  - Higher coverage
- TMR
  - 3rd copy is run only if error detected
  - Complex timing and synchronization
  - Voter
    - Single point of failure
- Effect on Overall System Reliability?

Summary

- Hardware ECC
  - Recommended for main memory when possible
- Software-Implemented ECC
  - Provide protection for code segments
  - Self-repair
- ARGOS Project
  - Continuous error collection
  - Automatic recovery
    - Increased availability

Future Work

- Observe Error Recoveries
  - Percentage of successful ones
  - Unpredicted cases
  - Improvements
- Reliability Analysis
  - Expand the “Scrub-Before-Use” Idea?

References