Fault Tolerance Software in ARGOS

by
Philip P. Shirvani
RATS
April 20, 1998

Outline

- Introduction
- System Features
- Requirements
- Error Detection Software
- Error Logging
- Summary

Introduction

- Collect Data
  - Processor errors in space
  - Hard vs. COTS board
  - Software error detection techniques
- Determine Tradeoffs
  - Fault avoidance
    - Radiation hardening, shielding, etc.
  - Fault tolerance
    - COTS components
    - Hardware and software techniques

ARGOS Experiment

Hardware Error Detection

- Rad-hard Board
  - Self-Checking Pair
  - EDAC for Main RAM
    - Memory scrubbing, SEU counter
    - Interface Between Chips Checked by Parity
    - Heartbeat
      - A counter increased by clock
  - Timers

Operating System

- VxWorks and System Routines
  - Multitasking
  - Memory load/dump; check-summed
- Task Synchronization and Message Passing
  - Semaphores, mailboxes
  - Blocking/non-blocking
- Dynamic Linking
Fault Tolerance Software in ARGOS

### Facilities
- **Diagnostic Tasks**
  - Test
    - Memory read/write
    - Cache read/write/invalidate
  - FPU
  - Timers
  - Maskable

### Requirements
- **Error Detection**
  - Exercise functional units
  - POST, SPEC benchmarks, DSP algorithms
- **Fault Tolerance**
  - Log errors safely
  - Correct Transmission of Log

### Constraints
- **Limited memory (2MB)**
  - Upload subset of programs
- **Compression**
- **Limited Communication Bandwidth**
  - Uplink: 1.1kbps
  - Downlink: 40kbps; 128kbps
  - 8 min. window every 101.6 min.
- **Readiness for Uplink**
  - Quick response time

### Software Error Detection
- **Stutter Step Mode Execution**
- **Control Flow Checking (SAI)**
- **Algorithm-Based Fault Tolerance (ABFT)**
- **Assertions**
- **Diversifying Values**
- **Watchdog Task**

### Diversifying Values
- **Avoid Common Values**
  - 0, 1, -1, small integers
- **Detect Error in Control Variables**
  - Loop counters + task ID
  - Flags + ID
- **Check Both Values of Flag**
  - Both ‘if’ and ‘else’ condition
- **Status Variables**
  - ‘enum’ → multiples of a prime

### Watchdog Timer
- **One Second Interrupt**
- **Maximum Execution Time**
- **Software Heartbeat**
  - Subroutine not locked
Fault Tolerance Software in ARGOS

Watchdog Task
- Simple Control Flow Checking
- Steps in a Subroutine
- General Watchdog
- Assigned Signature
- Combined with Timers

Sample Test

test1()
|
mark(TEST1_ID, T1_STEP1);
-
mark(TEST1_ID, T1_STEP2);
-
mark(TEST1_ID, T1_STEP3);
-
mark(TEST1_ID, T1_STEP4);
|

Sample Watchdog

```c
int MarkTable[MaxTests][MaxSteps];
int CurrentStep[MaxTests];

mark(int nTestId, int nStepNum)
|
| s = CurrentStep[nTestId];
| if (MarkTable[nTestId][s] != nStepNum)
| watchdog_error(nTestId, nStepNum);
| else
| | s++;
| | if (s == MaxStep[nTestId])
| | reset_watchdog(nTestId);
| | else
| | reset_timer(nTestId);
| |
```

Software Structure
- Main Control Program
- Functional Tests
- Error Handlers
- Error Logger
- Modular Design

Main Control Program
- Command Interpreter
  - Adding/deleting subroutines
  - Change parameters
- Calling/Forking Subroutines
  - Frequency controlled loops
- Logging Error
- Telemetry

Main Loop

```c
main()
|
| initialize();
| while (main_status != SHUTDOWN)
| |
| | process_ground_command();
| | run_next_test();
| | send_heartbeat();
| |
| | clean_up();
|
```
Fault Tolerance Software in ARGOS

Test Loops

```c
run_next_test()
{
  if (test_present[t] == TEST_P[t]) {
    for(i=TEST_ID[t]; i<TEST_ID[t]+test_freq[t]; i++) {
      if (test_health[t] == TEST_H[t])
        ((void*)TEST[t])();
      else if (test_health[t] == TEST_S[t])
        break;
      else
        data_error(LOCATION23);
    }
    if (i != TEST_ID[t]+test_freq[t])
      control_flow_error(LOCATION24);
  }
  t++; /* check with max */
}
```

Modular Design

- Dynamic Linking
- Upload Subroutines
  - New, update, repair
  - Guard send-message/function-call with health flags
  - Command to activate
- Advantages
  - Efficient Use of Bandwidth
  - Concurrent with Program Execution
- Version Numbers

Error Handlers

- Error Count (Most Important)
- Hardware Detections
  - e.g., double error in memory
- Software Detections
  - Detection mechanism
  - Caller ID
- Update Replicated Log

Error Logging

- Error Count (Most Important)
- Doubly Link List
  - Node Info
    - Time, type, detection mechanism, location, number, ID
  - Error detection and correction
  - Periodic check
- Duplicate
  - Multiple precision checksum

Fault Tolerance for Data

- Atomic Update
  - Keep old until commit
- Cross-Check
  - Redundant voters
- Different Format
  - Common-mode failure
  - Checksum formula
- Transmission
  - Send one copy with checksum
  - Acknowledge from ground

Replicated Critical Data

- Atomic Update
  - Keep old until commit
- Cross-Check
  - Redundant voters
- Different Format
  - Common-mode failure
  - Checksum formula
- Transmission
  - Send one copy with checksum
  - Acknowledge from ground
Fault Tolerance Software in ARGOS

Replica Manager
- Distributed Databases
- Commit protocols: 2PC, 3PC
- Separate Tasks
- Protocol
  - Make update on a copy with new version number
  - Exchange checksum
  - Exchange Go/No-Go
  - Commit (single operation) or retry

Miscellaneous Ideas
- Simplex Mode
- 32-bit Checksums
  - Full use of width
  - LFSR, MISR
- Copy Critical Data Across Boards
- Do Voting on Ground
- Code Checksum?

Summary
- Maximize Error Detection Coverage
- Modular Design
- Programming Practices
- Fault Tolerance for Critical Data
- Update Protocol

Future Work
- Coding and Debugging
- Refinements
- Checksum Formula
- Detection Techniques
- Fault Tolerance