Software Verification with a Model Checker

Ed Gamble, Gerard Holzmann

Laboratory for Reliable Software
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA

2009 Workshop on Spacecraft Flight Software
4 November 2009

Outline

- Key Points
- Analyzed Software Description
- Model Description
- Analysis Approach
- Results - Summary Table
- Directions
- Summary

Key Points

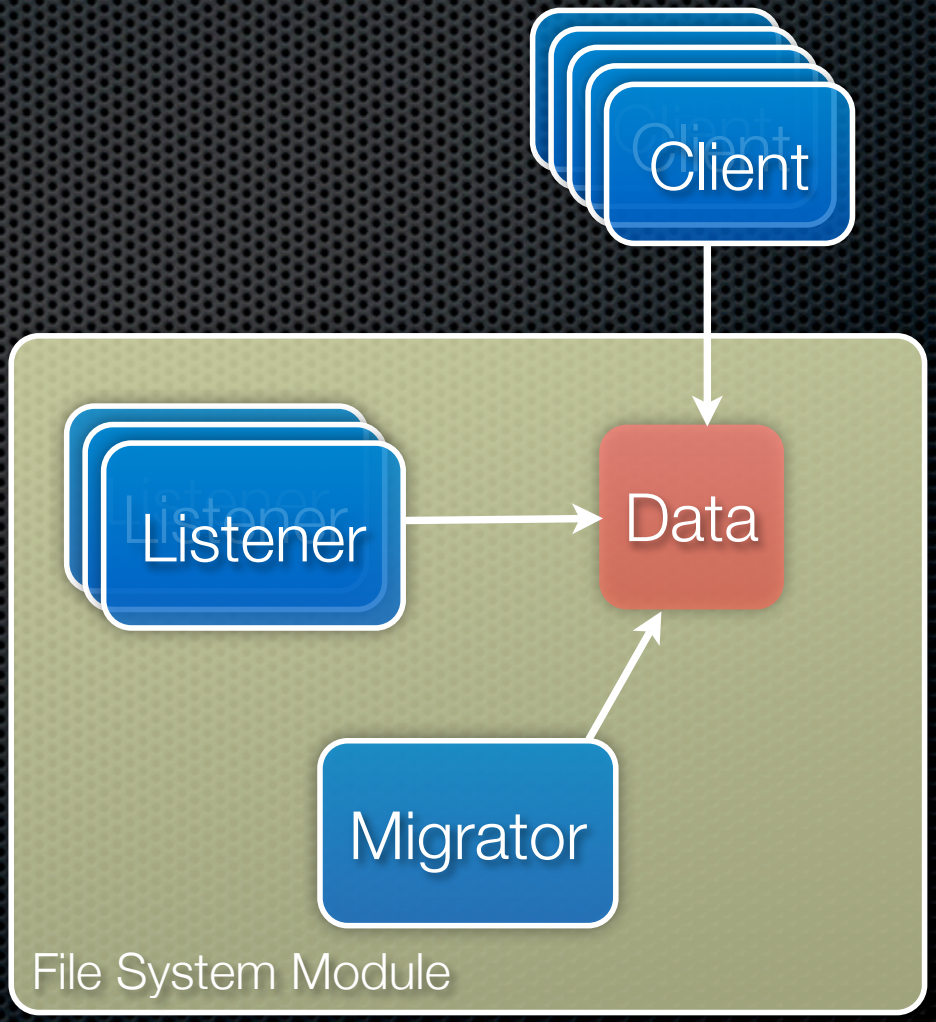
- Examined a massively parallel real-time file system.
- Designed a substantial Promela model.
- Analyzed using SPIN simulation and verification.
- Identified numerous errors.

Analyzed Software Description

- Overview: A massively parallel, real-time file system
- Purpose:
  - Provide hundreds of clients with persistent storage
  - Transparently allow ‘listeners’ to process (e.g. catalog, redirect) client data
- Design Philosophy: Be ‘lock free’ (ostensibly for performance reasons).

Analyzed Software Description

- Abstracted Architecture:
  - Hundreds of clients
  - Handful of ‘listeners’
  - Shared data structure
    - w/ both disjoint and overlapping data
  - Dynamic memory use



Model Description

- Capture the key functionality for:
  - multi-tasking operations upon the shared data
    - original motivation (as susceptible to errors)
  - dynamic memory management of file-system blocks
    - reclamation dependent on subtle state combinations between the multiple tasks

Model Description (p.2)

- Skip functionality for:
  - low-level memory allocation and supporting queues
    - experienced a false start w/ too much detail
  - full range of client interaction
    - unneeded given level of results thus far
    - can be added as needed

Model Implementation

- Promela used for implementation
  - ~1500 lines of code, 8 structure types, 5 arrays, 9 channels
  - 6 proctypes, 60 inlines
  - 50 assertions, >10 claims (and growing)
  - ~800 byte state-vector, >1000 x 10^6 states, > 1 x 10^6 depth
  - ➡ Not optimized; Not fully searchable within 32 GB

Model Implementation (p.2)

- Interplay between:
  - system behavior to be modeled
  - implementation details of the model
- Largely resolved by:
  - wrapping implementation details with ‘atomic’
  - meticulously following system logic, albeit abstracted

Model Implementation (p.3)

- Multiple model versions
  - Phase 1: Two clients, One Migrator
  - Phase 2: Two clients, One, Migrator, Two Listeners
    - Significant scope increase w/ notable overlaps
    - Model was updated to fix identified errors
    - Heavy use of assertions (to confirm implementation and system correctness)

Analysis Approach: Simulation

- Extensive use of SPIN simulation allowed for:
  - Learning the system's behavior
  - Confidence in the Promela implementation
    - Anomalous simulations could be attributed to model implementation errors or system errors.
      - High rate of system errors
      - ➡ Don't 'fix' the model behavior

Analysis Approach: Verification

- Two uses to confirm the:
  - model as suitably comprehensive
  - system itself (once simulation proved impractical)
    - Confirmed assertion violations found in simulation
    - Checked for non-progress (limited cases)
    - Minimized trails to facilitate problem analysis
    - Used ‘Swarm’ to identify errors quickly

Results: Summary

#	Type	Description	Source
1	logic	misplaced logic to identify reclaimable file-system block	Simulation
2	race	partially initialized data structure used before internally consistent	Simulation
3	array index	file-system block allocated prior to check for suitable space	Simulation
4	SPIN	backtracking error on array indexing assignment as: ‘array[array[index]] = value’	Verification
5	boundary	mishandled error case logic	Simulation
6	boundary	created, but empty, file can’t be reclaimed	Verification
7	race	value changed between conditional and alternate access: n = (n > x.val ? n : x.val)	Verification

Directions

- Unexplained assertion violations remain (or not!)
- Implement additional never claims
- Provide additional ‘listener’ tasks
- Embed ‘C’ code (from the system) in the SPIN model
- Optimize model to allow for complete verification

- Ascension of the mantra: “Safety Before Speed”

Summary

- A substantial Promela model was used to analyze a massively parallel real-time file system
  - Numerous errors identified
  - Notable interplay between model behavior and model implementation
  - Combined SPIN simulation and verification
  - Additional analysis options exist