Automated Application of Fault Tolerance Mechanisms in a Component-Based System

Isabella Thomm, Michael Stilkerich, Rüdiger Kapitza
Daniel Lohmann, Wolfgang Schröder-Preikschat

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Department of Computer Science 4
Distributed Systems and Operating Systems
Friedrich-Alexander University Erlangen-Nuremberg

http://www4.cs.fau.de/~isa
ithomm@cs.fau.de
Motivation

- Transient hardware faults become more likely
  - soft error rate in logic has increased by 9 orders of magnitude
  - soft error rate in SRAM is constantly high
  - soft errors cannot be ignored anymore
Motivation

- Transient hardware faults become more likely
  - soft error rate in logic has increased by 9 orders of magnitude
  - soft error rate in SRAM is constantly high
  - soft errors cannot be ignored anymore

- Hardware-based fault tolerance (FT) techniques
  - expensive: size, weight and power
Motivation

- Transient hardware faults become more likely
  - soft error rate in logic has increased by 9 orders of magnitude
  - soft error rate in SRAM is constantly high
  - soft errors cannot be ignored anymore

- Hardware-based fault tolerance (FT) techniques
  - expensive: size, weight and power

- Software-based fault tolerance (FT) techniques
Fault Tolerance - A Non-functional Property

- Techniques differ in protection properties and cost
- Measures are application specific

**Sw Fault Tolerance**

**Function Library**
- Seat Control
- Light System
- Wiper System
- ABS

**Diagram**

- Checking sums
- Redundancy
- Control Flow Monitoring
- FT Data Structures

- Configuration

**Variant A**
- Distributed System A
- Code

**Variant B**
- Distributed System B
- Code
Outline

- Motivation
- General Approach
- Prototype: Module Redundancy in KESO
- Conclusion and Future Work
Configurable Fault Tolerance

- Compiler-based approach
  - Separation functional code and fault tolerance
  - Plugins provide implementations of different FT techniques
  - Techniques can be combined
  - FT tailored to app, HW and safety requirements

- Automated FT application
  - Static system
  - Use of type-safe language
Application Model

- Fault Model: Transient hardware faults
- Mixed-criticality application
  - Safety-critical control application
  - Mapped to a periodic task
  - Sensible FT measure: module redundancy
- Requirements on application for module redundancy
  - Spatial and temporal isolation
  - Run-to-completion semantics, does not block
  - No side effects, e.g. no read from indeterministic sources
The KESO Multi-JVM

- **Domain as realm of protection**
  - Spatially isolated
  - Logical separation of heap, separate static fields

- **Portals for inter-domain communication**
  - Remote procedure call mechanism
  - Parameters are deep-copied

![Diagram showing control flows and ports for inter-domain communication](image)
FT Plugin Module Redundancy

- Example for one FT technique
  - Sensible in mixed-criticality systems

- Requirements
  - Configurability
  - Module Isolation
  - Replica Determinism
  - Automated Replication
Configurability

- Level of redundancy derived from safety requirements
- Sphere of replication is the domain
  - Replicated via configuration file
- Portal as transition point between single and replicated execution
  - Interface is identical for single and replicated mode
  - Parameters are deep-copied: own copy in each replica
Module Isolation

- **Spatial:** SW (domains) and HW-based (MPU) protection
  - physical separation of the domain heaps
- **Temporal:** Execution time budgets for task

![Diagram showing spatial and temporal isolation]
Replica Determinism

- Replica invocations must be ordered
- Critical locations *within* the replica executions
  - Communication (error spreading, indeterministic read of values)
  - Usually not in the scope of control applications
  - Problematic mechanisms are under control of the runtime system

Alternative:
- Implicit read and write access for external communication
Automated Replication

- Service: Java interface and implementation
- Replication plugin hooks into the proxy method
  - context switch
  - voter suitable for the return type and the number of replicas
  - recovery of a faulty domain from a healthy one
Conclusion

- Fault tolerance as a configurable property
  - FT is application specific
  - Balance protection and costs: Tailored runtime environment
  - Separation of FT measure from functional code
  - Compiler support

- Automated FT application dependent on
  - Application (written in type-safe language)
  - Hardware (Transient fault characteristics)
  - Safety requirements (amount of FT needed)

- KESO generates FT measure domain redundancy
  - Application does not have to be changed
  - Automated replication, generation of voting and recovery functions for a certain application
Future Work

- Additional FT techniques
  - Other replica and voting variants
  - Control flow information
  - Checksums to ensure data integrity, FT data structures, ...
- Hardening of runtime system
- Finer-grained hardening