Design Optimization of Fault-Tolerant Distributed Embedded Systems

Viacheslav Izosimov
Embedded Systems Laboratory (ESLAB)
Linköping University
Motivation

- Real-time applications
  - Time-constrained
  - Cost-constrained
  - Quality-of-service
  - **Fault-tolerant**
  - etc.

- Focus on **transient faults**
  and **intermittent faults**
# Transient and Intermittent Faults

## Transient faults
- Happen for a short time
- **Corruptions of data, miscalculation in logic**
- Do not cause a permanent damage of circuits
- Causes are outside system boundaries

## Intermittent faults
- Manifest similar as transient faults
- Happen repeatedly
- Causes are inside system boundaries
Transient and Intermittent Faults

Transient faults
- Radiation
- Lightning storms
- Electromagnetic interference (EMI)

Intermittent faults
- Internal EMI
- Power supply fluctuations
- Crosstalk
- Software errors (Heisenbugs)
Motivation

Transient faults are more likely to occur as:
- shrinking transistor size
- higher frequency
- lower voltage levels

Errors caused by transient faults have to be tolerated before they crash the system or lead to dramatic quality deterioration

However, fault tolerance against transient faults leads to significant performance and/or cost overhead
Our Contributions

- Scheduling with fault tolerance requirements
- **Fault tolerance policy assignment**
  - Combination of re-execution and replication
  - Checkpoint optimization
- Trading-off transparency for performance
- Mapping optimization with transparency
- **Mixed hard + soft** real-time fault-tolerant systems
  - Quasi-static scheduling algorithm
  - Introducing preemption into schedules
  - Trading-off between preemption and dropping
- **Hardware hardening vs. software fault tolerance**
  - Probabilistic fault model
  - Design optimization framework
System Architecture

**Processes:** Re-execution, Active Replication, Rollback Recovery with Checkpointing

**Messages:** Fault-tolerant predictable protocol

Maximum $k$ transient faults within each application run (system period)
Fault Tolerance Techniques

**Re-execution**

- Checkpoint overhead $\chi$
- Error-detection overhead $\alpha$
- Recovery overhead $\mu$

**Active replication**

$N_1$  $P_{1(1)}$  $N_1$  $P_{1(1)}$

$N_2$  $P_{1(2)}$  $N_2$  $P_{1(2)}$
Fault Tolerance Policy Assignment

Re-execution  Replication  Re-executed replicas

Viacheslav Izosimov, IDA/SaS/ESLAB 9  Fault-Tolerant Embedded Systems, 2009-08-20
Re-execution vs. Replication

Re-execution is better

Deadline
Missed

Met

Re-execution is better

Replication is better

Viacheslav Izosimov, IDA/SaS/ESLAB

Fault-Tolerant Embedded Systems, 2009-08-20
Fault Tolerance Policy Assignment

Viacheslav Izosimov, IDA/SaS/ESLAB

Fault-Tolerant Embedded Systems, 2009-08-20
Fault Tolerance Policy Assignment

Viacheslav Izosimov, IDA/SaS/ESLAB

Fault Tolerant Embedded Systems, 2009-08-20
Fault Tolerance Policy Assignment

Optimization
of fault tolerance
policy assignment
Experimental Results

Schedulability improvement under resource constraints

Mapping and replication ( MR )

Mapping and re-execution ( MX )

Mapping and policy assignment ( MRX )

Viacheslav Izosimov, IDA/SaS/ESLAB
Software-level Fault Tolerance

- Software fault tolerance
  - Reliability increase with time/space redundancy
  - Lead to lower performance
    - Fault tolerance overheads
    - Overheads due to error detection, voting, agreement
  - Low hardware cost

- Often cannot guarantee the required reliability levels and, at the same time, meet deadlines!
Hardening

- Improving the hardware architecture to reduce the error rate
  - Hardware redundancy (selective duplication of gates/units/nodes, dedicated additional hardware modules/flip-flops)
  - Re-designing the hardware to reduce susceptibility to transient faults
  - Using higher voltages / lower frequencies / larger transistor sizes
  - Protecting with shields

- Lead to lower performance
  - Use of technologies few generations back
  - Increase of the critical path and silicon area

- Very expensive
  - Extra-design effort / More expensive technologies
  - More silicon / Increase in the number of gates or computation units
  - Low production volumes

- Still may not guarantee the required reliability levels at affordable cost!
Motivation

Fault tolerance against transient faults may lead to significant performance or cost overhead!

Neither hardening nor pure software-level fault tolerance can guarantee the required level of reliability...

A trade-off between hardware and software fault tolerance has to be addressed to provide a reliable and low-cost system!
Architecture

Processes: Re-execution
Computation nodes: Hardening
Messages: Fault-tolerant predictable protocol

The error rates for each hardening version ($h$-version) of each computation node

The reliability goal $\rho = 1 - \gamma$

$\gamma$ is the maximum probability of a system failure due to transient faults on any computation node within a time unit.
Application Example

\[ \rho = 1 - 10^{-5} \]

Hardening versions of computation node \( N_1 \)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>N_1</strong></td>
<td>h = 1</td>
<td>h = 2</td>
</tr>
<tr>
<td>t</td>
<td>p</td>
<td>t</td>
</tr>
<tr>
<td>P_1</td>
<td>80</td>
<td>4 \cdot 10^{-2}</td>
</tr>
<tr>
<td>Cost</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

- \( t \) – worst-case execution time
- \( p \) – process failure probability
- Cost – \( h \)-version cost

Increase in reliability
Decrease in process failure probabilities
Application Example

\[ \rho = 1 - 10^{-5} \]

Worst-case execution times are increased

Hardening performance degradation (HPD)

Cost is increased with more hardening!

<table>
<thead>
<tr>
<th>( N_1 )</th>
<th>( h = 1 )</th>
<th>( h = 2 )</th>
<th>( h = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>( p )</td>
<td>( t )</td>
<td>( p )</td>
</tr>
<tr>
<td>( P_1 )</td>
<td>80</td>
<td>( 4 \cdot 10^{-2} )</td>
<td>100</td>
</tr>
<tr>
<td>Cost</td>
<td>10</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>

\( t \) – worst-case execution time
\( p \) – process failure probability
Cost – \( h \)-version cost
System Failure Probability (SFP) Analysis

We have proposed a system failure probability (SFP) analysis to connect error rates and the reliability goal to the number of re-executions in software.

\[ \rho = 1 - 10^{-5} \]

\[ T = 360\text{ms} \]

<table>
<thead>
<tr>
<th>( N_1 )</th>
<th>( h = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>( p )</td>
</tr>
<tr>
<td>( P_1 )</td>
<td>80</td>
</tr>
</tbody>
</table>

Non-trivial
Exact
Safe

main execution +

\[ k = 6 \text{ re-executions} \]
System Failure Probability (SFP) Analysis

We have proposed a system failure probability (SFP) analysis to connect error rates and the reliability goal to the number of re-executions in software.

\[ \rho = 1 - 10^{-5} \]

\[ T = 360\text{ms} \]

<table>
<thead>
<tr>
<th>( N_1 )</th>
<th>( h = 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>( p )</td>
</tr>
<tr>
<td>( P_1 )</td>
<td>100</td>
</tr>
</tbody>
</table>

main execution +

\[ k = 2 \text{ re-executions} \]
We have proposed a system failure probability (SFP) analysis to connect error rates and the reliability goal to the number of re-executions in software.

\[ \rho = 1 - 10^{-5} \]

\[ T = 360\text{ms} \]

<table>
<thead>
<tr>
<th>( N_1 )</th>
<th>( h = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>( p )</td>
</tr>
<tr>
<td>( P_1 )</td>
<td>160 ( \times 4 \cdot 10^{-6} )</td>
</tr>
</tbody>
</table>

\[ k = 1 \text{ re-executions} \]
Application Example

\[ \rho = 1 - 10^{-5} \]
\[ \mu = 20 \text{ ms} \]
\[ D = 360\text{ ms} \]

<table>
<thead>
<tr>
<th>( N_1 )</th>
<th>h = 1</th>
<th>h = 2</th>
<th>h = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>p</td>
<td>t</td>
<td>p</td>
</tr>
<tr>
<td>P_1</td>
<td>80</td>
<td>100</td>
<td>160</td>
</tr>
<tr>
<td>Cost</td>
<td>10</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>

\[ \rho = 1 - 10^{-5} \]
\[ \mu = 20 \text{ ms} \]
\[ D = 360\text{ ms} \]
Application Example

\[
\begin{align*}
D &= 360 \text{ ms} \\
\mu &= 15 \text{ ms} \\
\rho &= 1 - 10^{-5}
\end{align*}
\]

\[
\begin{align*}
N_1 & \quad N_2
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>(h = 1)</th>
<th>(h = 2)</th>
<th>(h = 3)</th>
<th>(h = 1)</th>
<th>(h = 2)</th>
<th>(h = 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_1)</td>
<td>(t)</td>
<td>(p)</td>
<td>(t)</td>
<td>(p)</td>
<td>(t)</td>
<td>(p)</td>
</tr>
<tr>
<td>(P_1)</td>
<td>60</td>
<td>(1.2 \cdot 10^{-3})</td>
<td>75</td>
<td>(1.2 \cdot 10^{-5})</td>
<td>90</td>
<td>(1.2 \cdot 10^{-10})</td>
</tr>
<tr>
<td>(P_2)</td>
<td>75</td>
<td>(1.3 \cdot 10^{-3})</td>
<td>90</td>
<td>(1.3 \cdot 10^{-5})</td>
<td>105</td>
<td>(1.3 \cdot 10^{-10})</td>
</tr>
<tr>
<td>(P_3)</td>
<td>60</td>
<td>(1.4 \cdot 10^{-3})</td>
<td>75</td>
<td>(1.4 \cdot 10^{-5})</td>
<td>90</td>
<td>(1.4 \cdot 10^{-10})</td>
</tr>
<tr>
<td>(P_4)</td>
<td>75</td>
<td>(1.6 \cdot 10^{-3})</td>
<td>90</td>
<td>(1.6 \cdot 10^{-5})</td>
<td>105</td>
<td>(1.6 \cdot 10^{-10})</td>
</tr>
<tr>
<td>Cost</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>Cost</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>
Application Example

N_1^2: P_1 P_3 P_{2/1} P_{2/2} P_{4/1} P_{4/2}

C_a = 32

N_2^2: P_1 P_3 P_{2/1} P_{2/2} P_{4/1} P_{4/2}

C_b = 40

N_1^3: P_1 P_3 P_2 P_4

C_c = 64

N_2^3: P_1 P_3 P_2 P_4

C_d = 80

N_1^2: P_1 P_{2/1} P_{2/2}

C_e = 72
Design Optimization Strategy

- Best Cost
- Meet Deadline
- Meet Deadline
- Satisfy Reliability
- SFP

Architecture Selection
Mapping
Hardening Setup
Number of Re-executions

- Architecture Optimization
- Mapping Optimization + Scheduling
- Hardening Optimization + Scheduling
- Re-execution Optimization (based on SFP)

Viacheslav Izosimov, IDA/SaS/ESLAB
Selected Experimental Results

**MAX** – hardware optimization  
**MIN** – software optimization  
**OPT** – combined architecture

**Accepted architecture:**
- Satisfying maximum accepted cost  
- Satisfying reliability goal  
- Schedulable

**Hardening performance degradation (HPD) 5%**
- Performance difference between the least hardened and the most hardened versions

**Maximum cost 20**

Viacheslav Izosimov, IDA/SaS/ESLAB

Fault-Tolerant Embedded Systems, 2009-08-20
Selected Experimental Results

**MAX** – hardware optimization
**MIN** – software optimization
**OPT** – combined architecture

Hardening performance degradation (HPD) 100%

Maximum cost 20

% accepted architectures as a function of soft error rate (SER)

Viacheslav Izosimov, IDA/SaS/ESLAB
Our Contributions

- Scheduling with fault tolerance requirements
- Fault tolerance policy assignment
  - Combination of re-execution and replication
  - Checkpoint optimization
- Trading-off transparency for performance
- Mapping optimization with transparency
- Mixed hard + **soft** real-time fault-tolerant systems
  - Quasi-static scheduling algorithm
  - Introducing preemption into schedules
  - Trading-off between preemption and dropping
- Hardware hardening vs. software fault tolerance
  - Probabilistic fault model
  - Design optimization framework
Thorough Design Optimization of Distributed Embedded Systems with Fault Tolerance is Essential