Fault-Tolerant Computing

Dealing with High-Level Impairments
About This Presentation

This presentation has been prepared for the graduate course ECE 257A (Fault-Tolerant Computing) by Behrooz Parhami, Professor of Electrical and Computer Engineering at University of California, Santa Barbara. The material contained herein can be used freely in classroom teaching or any other educational setting. Unauthorized uses are prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Nov. 2006</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Failure Confinement
"My goal is to be a failure. If I reach my goal, I'll feel successful and if I don't reach my goal, I'll feel successful too!"

"Remember son, if at first you don't succeed, assemble a team of lawyers to shield all involved, convince your investors that failure is an essential component of success, create a network for spin control to trivialize any negative consequences, and try, try again."
# Importance of Experimental Failure Data

- Indicate where effort is most needed
- Help with verification of analytic models

<table>
<thead>
<tr>
<th>System outage stats (%)*</th>
<th>Hardware</th>
<th>Software</th>
<th>Operations</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bellcore [Ali86]</td>
<td>26</td>
<td>30</td>
<td>44</td>
<td>--</td>
</tr>
<tr>
<td>Tandem [Gray87]</td>
<td>22</td>
<td>49</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>Northern Telecom</td>
<td>19</td>
<td>19</td>
<td>33</td>
<td>28</td>
</tr>
<tr>
<td>Japanese Commercial</td>
<td>36</td>
<td>40</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>Mainframe users</td>
<td>47</td>
<td>21</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td><strong>Overall average</strong></td>
<td><strong>30</strong></td>
<td><strong>32</strong></td>
<td><strong>24</strong></td>
<td><strong>14</strong></td>
</tr>
</tbody>
</table>

*Excluding scheduled maintenance

## Tandem unscheduled outages

- Power 53%
- Communication lines 22%
- Application software 10%
- File system 10%
- Hardware 5%

## Tandem outages due to hardware

- Disk storage 49%
- Communications 24%
- Processors 18%
- Wiring 9%
- Spare units 1%
Failure Data Used to Validate or Tune Models

- Indicate accuracy of model predictions (compare multiple models?)
- Help in fine-tuning of models to better match the observed behavior

**Example:** Two disks, each with MTTF = 50,000 hr, MTTR = 5 hr

Disk pair failure rate \( \approx \frac{2\lambda^2}{\mu} \)

Disk pair MTTF \( \approx \frac{\mu}{2\lambda^2} \)

\[ \approx 2.5 \times 10^8 \text{ hr} = 285 \text{ centuries} \]

In 48,000 years of observation (2 years \( \times \) 6000 systems \( \times \) 4 disk pairs), 35 double disk failures were reported \( \Rightarrow \) MTTF \( \approx 14 \) centuries

Problems with experimental failure data:
- Difficult to collect, while ensuring uniform operating conditions
- Logs may not be complete or accurate (the embarrassment factor)
- Assigning a cause to each failure not an easy task
- Even after collection, vendors may not be willing to share data
- Impossible to do for one-of-a-kind or very limited systems
Preparing for Failure

- Minimum requirement: accurate estimation of failure probability
- Putting in place procedures for dealing with failures when they occur

Failure probability = Unreliability

Reliability models are by nature pessimistic (provide lower bounds)
However, we do not want them to be too pessimistic

\[
\text{Risk} = \frac{\text{Consequence / Unit time}}{\text{Events / Unit time}} \times \frac{\text{Consequence / Event}}{\text{Magnitude}}
\]

Frequency may be equated with unreliability (failure probability)
Magnitude is estimated via economic analysis

Failure handling is often the most neglected part of the process
An important beginning: clean, unambiguous messages to operator/user
Listing the options and urgency of various actions is a good idea
Two way system-user communication (adding user feedback) helpful
Quality of failure handling affects the “Magnitude” term in risk equation
Very Small Probabilities: The Human Factor

Interpretation of data, understanding of probabilities, acceptance of risk

Risk of death / person / year
Influenza 1/5K
Struck by auto 1/20K
Tornado (US MW) 1/455K
Earthquake (CA) 1/588K
Nuclear power plant 1/10M
Meteorite 1/100B

US causes of death / 10^6 persons
Auto accident 210
Work accident 150
Homicide 93
Fall 74
Drowning 37
Fire 30
Poisoning 17
Civil aviation 0.8
Tornado 0.4
Bite / sting 0.2

Factors that increase risk of death by 1/10^6 (deemed acceptable risk)
Smoking 1.4 cigarettes
Drinking 0.5 liter of wine
Biking 10 miles
Driving 300 miles
Flying 1000 miles
Taking a chest X-ray
Eating 100 steaks

Risk underestimation factors:
Familiarity, being part of our job, remoteness in time or space

Risk overestimation factors:
Scale (1000s killed), proximity
Believability and Helpfulness of Failure Warning

“No warning system will function effectively if its messages, however logically arrived at, are ignored, disbelieved, or lead to inappropriate actions.” Foster, H.D., “Disaster Warning Systems,” 1987

**Unbelievable failure warnings:**
- Failure event after numerous false alarms
- Real failure occurring in the proximity of a scheduled test run
- Users or operators inadequately trained (May 1960 Tsunami in Hilo, Hawaii, killing 61, despite 10-hour advance warning via sirens)

**Unhelpful failure warnings:**
- Autos – “Check engine” indicator
- Computer systems – “Fatal error” message
Engineering Ethics

Risks must be evaluated thoroughly and truthfully

**IEEE Code of Ethics:** As IEEE members, we agree to

1. accept responsibility in making decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment

6. maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;

7. seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others

**ACM Code of Ethics:** Computing professionals must minimize malfunctions by following generally accepted standards for system design and testing

give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks
Speed of Failure Detection

Prompt failure detection is a prerequisite to failure confinement

In many cases dealing with mechanical elements, such as wing flaps, reaction time of 10s/100s of milliseconds is adequate (reason: inertia)

In some ways, catastrophic failures that are readily identified may be better than subtle failures that escape detection

Example: For redundant disks with two-way mirroring, detection latency was found to have a significant effect on the probability of data loss


Failure detection latency can be made negative via “failure prediction” (e.g., in a storage server, increased error rate signals impending failure)
Fail-Safe Systems

**Fail-safe:** Produces one of a predetermined set of safe outputs when it fails as a result of “undesirable events” that it cannot tolerate

Fail-safe traffic light: Will remain stuck on red

Fail-safe gas range/furnace pilot: Cooling off of the pilot assembly due to the flame going out will shut off the gas intake valve

A fail-safe digital system must have at least two binary output lines, together representing the normal outputs and the safe failure condition

Reason: If we have a single output line, then even if one value (say, 0) is inherently safe, the output stuck at the other value would be unsafe

Two-rail encoding is a possible choice: 0: 01, 1: 10, F: 00, 11, or both

**Totally fail-safe:** Only safe erroneous outputs are produced, provided another failure does not occur before detection of the current one

**Ultimate fail-safe:** Only safe erroneous output is produced, forever
Fail-Safe System Specification

Amusement park train safety system

Signal $s_B$ when asserted indicates that the train is at beginning of its track (can move forward, but should not be allowed to go back)

Signal $s_E$ when asserted indicates that the train is at end of its track (can go back, but should not move forward)

Is the specification above consistent and complete?

No, because it does not say what happens if $s_B = s_E = 1$; this would not occur under normal conditions, but because such sensors are often designed to fail in the safe mode, the combination is not impossible

Why is this a problem, though? (Train simply cannot be moved at all)

Completeness will prevent potential implementation or safety problems
Fail-Safe 2-out-of-4 Code Checker

Input: 4 bits \textit{abcd}, exactly 2 of which must be 1s

Output: \(fg = 01\) or \(10\), if the input is valid
- 00 safe erroneous output
- 11 unsafe erroneous output

Output will become permanently 00 upon the first unsafe condition
Fail-Safe State Machines

Use an error code to encode states
Implement the next-state logic so that the machine is forced to an error state when something goes wrong

Possible design methodology:
Use Berger code for states, avoiding the all 0s state with all-1s check, and vice versa
Implement next-state logic equations in sum-of-products form for the main state bits and in product-of-sums form for the check state bits
The resulting state machine will be fail-safe under unidirectional errors

<table>
<thead>
<tr>
<th>State</th>
<th>Input (x=0)</th>
<th>Input (x=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>A</td>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>001 10</td>
</tr>
<tr>
<td>B</td>
<td>010 10</td>
</tr>
<tr>
<td>C</td>
<td>011 01</td>
</tr>
<tr>
<td>D</td>
<td>100 10</td>
</tr>
<tr>
<td>E</td>
<td>101 01</td>
</tr>
</tbody>
</table>

Hardware overhead for \(n\)-state machine consists of \(O(\log\log n)\) additional state bits and associated next-state logic, and a Berger code checker connected to state FFs
Principles of Safety Engineering

**Principles for designing a safe system** (J. Goldberg, 1987)

1. Use barriers and interlocks to constrain access to critical system resources or states

2. Perform critical actions incrementally, rather than in a single step

3. Dynamically modify system goals to avoid or mitigate damages

4. Manage the resources needed to deal with a safety crisis, so that enough will be available in an emergency

5. Exercise all critical functions and safety features regularly to assess and maintain their viability

6. Design the operator interface to provide the information and power needed to deal with exceptions

7. Defend the system against malicious attacks
Recovery from Failures

The recovery block scheme (originally developed for software)

<table>
<thead>
<tr>
<th>ensure by</th>
<th>acceptance test</th>
</tr>
</thead>
<tbody>
<tr>
<td>by</td>
<td>primary module</td>
</tr>
<tr>
<td>else by</td>
<td>first alternate</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>else by</td>
<td>last alternate</td>
</tr>
<tr>
<td>else fail</td>
<td></td>
</tr>
</tbody>
</table>

e.g., sorted list
e.g., quicksort
e.g., bubblesort

Computer system with manual backup may be viewed as a one-alternate recovery block scheme, with human judgment constituting the acceptance test.

Instead of resorting to an alternate (hardware/software) module, one may reuse the primary one.

This scheme is known as “retry” or time redundancy and is particularly effective for dealing with transient or soft failures.
Fail-Stop and Failover Strategies

**Fail-stop systems:** Systems designed and built in such a way that they cease to respond or take any action upon internal malfunction detection. Such systems do not confuse the users or other parts of a distributed system by behaving randomly or, worse, maliciously upon failure. A subclass of fail-safe systems (here, stopping is deemed a safe output).

**Failover:** Upon failure detection, often of the fail-stop kind, requests and other tasks are redirected to a back-up system.

Example – Failover on long-running connections for streaming media: all that is needed is to know the file being streamed, and current location within the file, to successfully switch over to a different server.

Failover software is available for Web servers as part of firewalls for most popular operating systems.

It monitors resources and directs requests to a functioning server.