Recovering from Errors

- Two basic approaches
  - Forward Error Recovery (FER)
  - Backward Error Recovery (BER)
- FER: continue to go forward in presence of errors
  - Use redundancy to mask effects of errors
  - E.g., have a co-pilot that can seamlessly take over airplane
- BER: go backward to recover from errors
  - Use redundancy to enable recovery to saved good state of system
  - E.g., go back to old saved version of file that you corrupted

Forward Error Recovery

- Canonical example: triple modular redundancy (TMR)
  - Majority voter chooses correct output
  - Masks error in any one of the three modules
Backward Error Recovery

• Canonical examples
  – Periodic checkpoint/recovery
  – Logging of changes to system state

• BER designs tend to be more complicated
  – We’ll spend more time on them later in this part of the course

Very Rough Comparison: FER vs. BER

<table>
<thead>
<tr>
<th>Feature</th>
<th>FER</th>
<th>BER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault-free performance</td>
<td>Some degradation</td>
<td>Little degradation</td>
</tr>
<tr>
<td>Performance if faults</td>
<td>No slowdown</td>
<td>Slow recovery</td>
</tr>
<tr>
<td>Hardware cost</td>
<td>Higher</td>
<td>Lower</td>
</tr>
<tr>
<td>Design complexity</td>
<td>Lower</td>
<td>Higher</td>
</tr>
</tbody>
</table>

Performance of FER vs. BER

Warning: do not take this graph too seriously. The relative heights of the curves and their shapes are gross estimates that do not correspond to any particular system.

System Design Space

Systems tend to get only 2 out of 3 features

- Backward Error Recovery
- Forward Error Recovery
- PCs and laptops

High Availability

Low Cost

High Performance
Types of Redundancy

• Physical (spatial)

• Temporal

• Design

• Information

Physical (Spatial) Redundancy

• Physically replicate a module
  – Most obvious approach

• Design issues
  – How many replicas are needed?
    » For error detection?
    » For error correction?
  – How are errors detected/corrected?
  – Is the redundancy “active” or “passive”?

• Canonical example: triple modular redundancy (TMR)
  – 3 replicas
  – Errors corrected by majority voter
  – Redundancy is passive (no special action taken if error detected)

Temporal Redundancy

• Replicate the actions on a module using the same module, but at a different time

• Effective for tolerating transient faults
  – Can this help at all for hard faults?

• We’re going to spend more time on this later!

Design Redundancy

• Use multiple different designs to guard against a fault in any of them

• Disadvantage: very costly to have multiple different designs
  – Also: doesn’t tolerate faults in design specification – only tolerates faults in implementations of that specification

• Examples
  – “N-version programming” – have N design teams develop N different versions of a piece of software

• We’ll come back to this later in the course, mostly in the context of software fault tolerance
Information Redundancy

- For a given k-bit piece of information, add r check bits to it that make it possible to detect/correct errors in the original k-bit information
- Example: parity bit
  - By adding a single bit to a word of information, we can detect any single-bit error in it
- Example: checksum
  - Sender computes a number (checksum) that nearly uniquely identifies long stream of information that it sends, and it sends checksum along with information
  - Receiver uses same algorithm to compute checksum on information it receives (not including checksum). If receiver's checksum matches sender's checksum, then information is error-free
- We'll talk much more about this soon

Outline

- Basic Concepts
  - Physical Redundancy
  - Error Detecting/Correcting Codes
  - Re-Execution Techniques
  - Backward Error Recovery Techniques

Physical Redundancy: TMR

- Strengths
  - Tolerates an error in any single module
  - Tolerates soft and hard errors
  - Simple design
  - Small performance penalty, even when faults occur
- Weaknesses
  - Can't tolerate multiple faults
    - Can't tolerate any faults after a latent hard fault
  - Expensive hardware (3x cost)
  - Uses lots of power (approx 3x power of unprotected)
  - Single point of failure at voter
  - Can't tolerate errors due to design faults ... why not?

Physical Redundancy: NMR

- N-modular redundancy (N is an odd integer)
  - Why is N odd?
  - Can tolerate more errors than TMR
    - Tolerates up to N/2 = ½ errors
- Cost = N*cost of module
- Still has single point of failure at voter!
  - But voter is simple and can be designed to be very robust
- One solution to single voter problem
  - "Restoring organ" = TMR with triplicated voter
  - How does this help?
Physical Redundancy: Boeing 777

• Boeing 777 requires near-perfect reliability
• Its main flight computer:
  – Has 3 identical units in a TMR configuration
  – Each of these units has 3 processors in a TMR configuration
  – The three processors in each unit are heterogeneous:
    » Intel 80486 (the x86 before the original Pentium)
    » Motorola 68040
    » AMD 29050

Called “triple-triple redundant 777 primary flight computer” in paper by Yeh, 1996.

Physical Redundancy: Active vs. Passive

• NMR is passive, since the system doesn’t really do anything different when an error is detected
• With active redundancy, the system detects the error, locates/diagnoses it, and reconfigures to tolerate it

• Standby sparing has at least two redundant modules
  – Operational module has internal error detection mechanism
  – If a hard error is detected, the system reconfigures to use a spare
    » “Cold standby”: standby was inactive and must be warmed up
    » “Hot standby”: standby was active and is in correct state

More Active Redundancy

• Pair-and-spare
  – Like standby sparing, except each module is a pair
  – This pair compares outputs to detect errors
  – If error detected, a spare module (i.e., pair) is configured in

Hybrid Physical Redundancy

• Combine passive and active redundancy
• Example: NMR with spares
  – Let’s say we have 5 replicas
  – Organize 3 into a TMR scheme
  – Save other 2 for use as spares
Hybrid Physical Redundancy

• Combine passive and active redundancy
• Example: NMR with spares
  – Let’s say we have 5 replicas
  – Organize 3 into a TMR scheme
  – Save other 2 for use as spares
  – After first hard fault, map in a spare

  V

Watchdog Timers

• So far, we’ve figured out how to detect when something is wrong … but how do we detect when we’re not doing anything at all?
• Watchdog timer monitors a module and triggers a recovery if the module doesn’t do anything in a given amount of time
  – E.g., put a watchdog timer on a microprocessor bus
• Who watches the watchdog?
  – If we assume single fault scenario, then this usually isn’t a problem
  – But what if watchdog has hard fault that causes it to never timeout and trigger a recovery?

The Teramac

• “The Teramac Custom Computer: Extending the Limits with Defect Tolerance” (Culbertson et al.)