Curriculum Development Centers Program

Awardee of The Office of the National Coordinator for Health Information Technology

Component 8
Installation and Maintenance of Health IT Systems

Unit 9a

Creating Fault-Tolerant Systems, Backups, and Decommissioning

This material was developed by Duke University, funded by the Department of Health and Human Services, Office of the National Coordinator for Health Information Technology under Award Number IU24OC000024.

What We'll Cover

- What is fault tolerance?
- Why are redundancy and fault tolerance Important?
- Three levels of fault tolerance
- Six rules of fault tolerance in a system
- Getting technical: creating fault tolerance
- Backup strategies
- Tips on decommissioning data & hardware

Redundancy and Fault Tolerance

- Dependence on EHRs is increasing exponentially.
- EHR systems require redundant, or "failover", resources and fault tolerance to ensure uptime and data integrity.
- "Failure" vs "fault": fault is the cause of a failure of the system to comply with its specifications / precise requirements.
- Ask vendor how fault tolerance is designed/coded into the EHR application.

Redundancy and Fault Tolerance (cont'd)

- Forrester Consulting report (2010) on server failure during prior two years:
  - ¾ experienced downtime.
  - Only 1% of server outages were resolved within five minutes.
  - 68% had impact on clinical activities.
  - 50+% affected administrative processes.

Three Levels of Fault Tolerance

1. Hardware fault tolerance
   - Extra hardware resources
   - E.g., redundant communications, replicated processors, additional memory, redundant power/energy supplies
2. Software fault tolerance
   - Compensating for faults such as changes in program or data structures due to transients or design errors
   - E.g., checkpoint/restart, recovery blocks, multiple-version programs
3. System fault tolerance
   - Compensating for failures in other system facilities that are not computer-based
   - E.g., software to detect & compensate for sensor failure

Six Rules of Fault Tolerance

In "A Conceptual Framework for Systems Fault Tolerance", 6 rules are outlined:

- **Rule 1:** Know precisely what the system is supposed to do.
  - How long can system be allowed to deviate from specifications before being declared a "failure"?
  - What abnormal conditions must be accommodated?
- **Rule 2**: Look at what can go wrong.
  - Group causes into classes.
  - Define "fault floor".

Six Rules of Fault Tolerance (cont'd)

- **Rule 3:** Study your application & determine appropriate fault containment regions & earliest feasible time to deal with potential faults.
  - Fault tolerance generally means more resources (time & space)
- **Rule 4**: Completely understand application requirements & use them to make appropriate time/space trade-offs.
  - Consider costs, & classify faults by likelihood.

Six Rules of Fault Tolerance (cont'd)

- **Rule 5**: Concentrate on credible faults first.
  - Ignore less likely faults unless they require little additional cost. Mitigate the most likely faults first.
- **Rule 6**: Determine application failure margins.
  - Balance the degree of fault tolerance needed with the cost of implementation.

Creating Fault Tolerance: Hardware

- Features: hot-add memory, hot-swappable hard drives, hot-plug PCI-X slots (add/remove PCI expansion cards), redundant power supplies & cooling fans
- Choose fault-tolerant servers over clustered servers (less reliable & more difficult to maintain).
- Measure ROI (return on investment) against costs of downtime: safety, lost productivity, financial, litigation, disruption.
- Mirror critical systems & disperse throughout the network. Consider hot spare servers.

Creating Fault Tolerance: Data Storage

- RAID (Redundant Array of Independent Disks)
  - Available on systems where basic disks have been changed to dynamic disks
  - RAID 1 (disk mirroring): fault tolerance for boot/system volumes
  - RAID 5 (disk striping with parity): increased speed & reliability for high-transaction data volumes, such as those hosting databases
  - Hardware RAID generally higher performance but more expensive than software RAID.
  - Distributed File System (DFS), File Replication Service (FRS)

Creating Fault Tolerance: VSS

- Windows Volume Shadow Copy Service (VSS)
  - Keeps point-in-time snapshots of data volumes.
  - Users can recover accidentally deleted files or revert to earlier versions of documents.

Creating Fault Tolerance: Virtualization

- Server virtualization: multiple virtual operating systems run on single physical machine yet remain logically distinct.
- Advantages: single environment & license; protection; redundancy (no single point of failure); flexible in storage type; basic system management skills needed; supports applications without modification; simple; less expensive
- Consider combining with duplicate hardware hosting.
- Limitations: some programs don't run well in virtual environment, e.g. frequent memory access.

Creating Fault Tolerance: System-Wide

- Distributed architecture: maintains access to application despite network interruption.
- Uninterruptible Power Supply (UPS) & backup power in key areas, e.g. server rooms, wiring closets
- Redundancy & fault tolerance in network infrastructure switches, routers, & WAN links: provides secondary network connections between sites.

Creating Fault Tolerance: System-Wide (cont'd)

- Windows networks: consider Network Load Balancing (NLB).
  - Scales application to run on up to 32 separate servers, increases availability.
  - Provides fault tolerance through failover support for applications and services running on IP networks.

Summary

- A *failure* is defined as deviating from compliance with the system *specification*. When delivering a service to the user
- A *fault* is the adjudged cause of a failure.
- Systems that have fault tolerance/ redundancy built into their hardware and/ or software to minimize downtime, even during a failure.

Summary

- Six rules to follow When developing a fault tolerance strategy:
  - Know precisely what the system is supposed to do.
  - Look at what can go wrong.
  - Study your application(s)
  - Completely understand application requirements & use them to make appropriate time/space trade-offs
  - Concentrate on credible faults first
  - Determine application failure margins

Summary

- Consider:
  - Using hardware designed with special features for fault tolerance
  - Consider the ROI against costs associated with downtime/ safety concerns
  - Consider hardware/ network diversification over consolidation when focusing on fault tolerance including hardware virtualization
  - Special considerations for data storage including RAID

References

- NIST. “A Conceptual Framework for System Fault Tolerance.”
  - http://hissa.nist.gov/chissa/SEI_Framework/framework_20.html
  - http://hissa.ncsl.nist.gov/chissa/SEI_Framework/framework_3.html
- Mitch Tuloch. “Implementing Fault Tolerance on Windows Networks”.
  - http://www.windowsnetworking.com/articles_tutorials/Implementing-Fault-Tolerance-Windows-Networks.html
- Roy Sanford. “Electronic Health Records Need a Fail-Proof Foundation to Deliver on Quality, Economy Promises.” *Health News Digest*, Apr. 5, 2010.
  - http://www.healthnewsdigest.com/news/Guest_Columnist_710/Electronic_Health_Records_Need_a_Fail-Proof_Foundation_to_Deliver_on_Quality_Economy_Promises_2_printer.shtml