



# ACCELERATE

outcomes with convergence

HP **TECHFORUM** 2010

CONVERGE

INNOVATE

TRANSFORM

# Maximizing High Availability

Keith Parris

Systems/Software Engineer, HP

June 23 2010





# AVAILABILITY METRICS

## – Concept of “Nines”

Availability	Availability (%)	# Nines	Downtime annually	In units of
0.99	99%	2	3.6525	days
0.999	99.9%	3	8.766	hours
0.9999	99.99%	4	52.596	minutes
0.99999	99.999%	5	315.576	seconds



# KNOW THE MOST FAILURE-PRONE COMPONENTS: HARDWARE

- Things with moving parts
  - Fans
  - Spinning Disks
- Things which generate a lot of heat
  - Power supplies
- Things which are very new (not yet “burned in”)
  - “Infant Mortality” of electronics
- Things which are very old
  - Worn out



# KNOW THE MOST FAILURE-PRONE COMPONENTS: SOFTWARE

- Newly-written software
  - V1.0
- New features in existing software
- Recent fixes for bugs
- Seldom-used code paths
  - Error-handling or recovery code
  - Previously-unused features
- Old code which has worked for years, but around which the environment has changed recently in some way



# CONSIDER FACILITIES AND INFRASTRUCTURE

## – Datacenter needs:

- UPS
- Generator
- Reserve capacity for A/C
- Power from two substations
- Redundant UPS, Generator
- Dual power connections to equipment
- Redundant cooling



# CONSIDER FACILITIES AND INFRASTRUCTURE

- The Uptime Institute (<http://uptimeinstitute.org>) has produced a white paper which classifies datacenters into 4 Tiers:
  - Tier I (Basic)
  - Tier II (Redundant Capacity Components)
  - Tier III (Concurrently Maintainable)
  - Tier IV (Fault Tolerant and Concurrently Maintainable)
    - [http://www.uptimeinstitute.org/wp\\_pdf/\(TUI3026E\)TierClassificationsDefineSiteInfrastructure.pdf](http://www.uptimeinstitute.org/wp_pdf/(TUI3026E)TierClassificationsDefineSiteInfrastructure.pdf)



# CONSIDER FACILITIES AND INFRASTRUCTURE

- The Uptime Institute white paper says **“Even a Fault Tolerant and Concurrently Maintainable Tier IV site will not satisfy an IT requirement of Five Nines (99.999 percent) uptime.”**
  - So a datacenter with Best Practices in place can do “4 Nines” at best



# CONSIDER FACILITIES AND INFRASTRUCTURE

- Conclusion: You need multiple datacenters to go beyond “4 Nines”
- The Uptime Institute white paper says
  - **“Clients with very high availability requirements for their end users often consider multiple sites working in concert. This requires a more complex IT architecture and networking configuration but will provide the ability to have the unfortunate EPO (Emergency Power Off) or fire event at one site be transparent to the end users.”**



# ELIMINATE SINGLE POINTS OF FAILURE

- Can't depend on anything of which there is only one
- Incorporate redundancy
- Examples:
  - DNS Server at only one site in Disaster-Tolerant Cluster
  - Single storage box under water sprinkler



# ELIMINATE SINGLE POINTS OF FAILURE

- Cover the most-often-failing components first, using redundancy
- Also consider consequences of failure of **any** component, however unlikely
  - Even things thought to be very reliable and proven can still sometimes fail
- Design to accommodate a failure of **any** component



# ELIMINATE SINGLE POINTS OF FAILURE

- Insufficient capacity can also represent a SPOF
  - e.g. A/C
- Inadequate performance can also adversely affect availability
- Cross-train staff so no individual person becomes a SPOF



# ELIMINATE SINGLE POINTS OF FAILURE

- Even an entire geographically-dispersed disaster-tolerant cluster could be considered to be a potential SPOF
- HP Reliable Transaction Router (RTR) Software can route transactions to two different back-end servers (& each could be a DT cluster)
- Some customers have provisions in place to move customers (and their data) between two different clusters



# DETECT AND REPAIR FAILURES RAPIDLY

- MTBF and MTTR both affect availability

- Redundancy needs to be maintained by repairs to prevent outages from being caused by subsequent failures

MTBF

- Availability = -----

(MTBF + MTTR)

- Mean Time Between Failures quantifies how likely something is to fail
- Mean Time To Repair quantifies how quickly a problem can be resolved, and it affects the chances of a second failure causing downtime

- We must detect failures quickly so repairs can be initiated quickly

- This requires proactive monitoring



# TAKE ADVANTAGE OF INCREASED REDUNDANCY LEVELS

- If the technology supports it, consider going beyond 2X redundancy to 3X, 4X, or even higher levels of redundancy
- Multiple-site Disaster Tolerant clusters
  - Protect against loss of entire datacenter sites; even loss of multiple entire datacenter sites



# SOLVE PROBLEMS WHICH OCCUR ONCE AND FOR ALL

- Record data about failures so they can be solved, and not occur repeatedly
  - Log errors and events
  - Put Console Management system in place to log serial console output
  - Take crash dumps
- Log service calls; follow up on them
- Insist on getting fixes for problems



# AVOID PROBLEMS

- Ensure that documentation is accurate and up-to-date
- Ensure that equipment and cables are fully and correctly labeled



# RECORD DATA AND HAVE TOOLS IN PLACE IN CASE YOU NEED THEM

- Error and Event Logs

- Synchronize clocks on all equipment using NTP so logs & events are in synch

- Performance Data



# CONSIDER COMPONENT MATURITY

- Reliability of products tends to get better over time as improvements are made
  - ECOs, patches, quality improvements
- Common advice is “Never use V1.0 of anything”



# CONSIDER COMPONENT MATURITY

- Don't be the first to use a new product or technology
  - Not enough accumulated experience in the field yet to find and correct the problems
- If it's working well, don't throw out a perfectly good "mature" product just because it's not fashionable anymore
- Don't be the last to continue using an old product or technology past the end of its market life
  - Support eventually goes away, and support quality certainly degrades over time as memory fades



# IMPLEMENT A TEST ENVIRONMENT

- Duplicate production environment as closely as possible
  - Run your unique applications
  - Same exact HW/SW/FW if possible
  - Same scale if possible
- Test new things first in the Test Environment, not in the Production Environment
- May be able to leverage this equipment for Development / Quality Assurance / Disaster Recovery to help justify cost



# SOFTWARE VERSIONS AND FIRMWARE REVISIONS

- Stay on supported versions
  - Troubleshooting assistance is available
  - Problems you encounter will get fixed
- Don't be the first to run a new version
- Don't be the last to upgrade to a given version (support personnel tend to forget details of issues on ancient releases)
- Test new versions in a test environment first



# PATCHES AND ECO KITS

- Monitor patch/ECO kit releases
  - Know what potential problems have been found and fixed
  - Consider the likelihood you will encounter a given problem before installing the associated patch
- If you require high availability, consider letting a patch “age” a while after release before installing it in production
  - Avoids installing a patch only to find out it’s been recalled due to a problem



# PATCHES AND ECO KITS

- Install patches/ECO kits in test environment to detect problems with the fix in your specific environment
- Avoid waiting too long to install patches/ECO kits, or you may suffer a problem which has already been fixed



# MANAGING CHANGE

- Changes in either the configuration or environment introduce risks
- Change is unavoidable; changes in the environment around the system will dictate that changes occur within the system to keep down risks
- Change must be managed, not prevented
- There must be a trade-off and a balance between keeping up-to-date and minimizing unnecessary change



# MANAGING CHANGE

- New features often introduce new bugs
- Fixes to problems can themselves cause new problems
- Try to change only one thing at a time
  - Makes it easier to determine cause of problems and makes it easier to back out changes
- Each change proposed should include a documented, tested back-out procedure
  - In very rare cases, the nature of a change is such that it actually cannot be backed out once implemented; in this case, this fact needs to be documented, the additional risk recognized, and preparations put in place to fix any problems moving forward
- Always test any changes first in a test environment



# MANAGING CHANGE

- Plan for regular, periodic scheduled downtime for various maintenance activities
- You **WILL** have downtime. Your choice is whether you want it to be mostly ***scheduled***, or ***unscheduled*** downtime.



# MANAGING CHANGE

- When a new problem occurs, it is common to scrutinize recent changes to see if they could possibly have caused or contributed to the problem
  - So change tracking is a valuable part of change management
- But keep in mind that things outside the control of the change management process can also cause problems



# COMPUTER APPLICATION OF BIODIVERSITY

- Using a mix of different implementations improves survivability
- Examples:
  - British Telecom's X.25 network
  - DE500 and DE602 LAN adapters



# MINIMIZE COMPLEXITY, AND THE NUMBER OF THINGS WHICH CAN FAIL

- Try to minimize the number of things which could fail and cause an outage or disruption
- Examples:
  - Simplest hardware necessary for a task
  - Minimize number of tiers for a service



# REDUCE THE EXTENT OF IMPACT OF A FAILURE

- Popular adage: “Don’t put all your eggs in the same basket”
- Minimize the effect of any single failure. Techniques:
  - Preserve the independence of different resources
  - Division of services into portions so that failure of one portion doesn’t affect all portions
    - This is a trade-off between span of impact, and complexity and thus frequency of failures



# REDUCE THE EXTENT OF IMPACT OF A FAILURE

- Be cautious about sharing resources among unrelated services, despite the lure of cost savings
  - Shared storage subsystems, storage virtualization
  - Server virtualization



# REDUCE THE EXTENT OF IMPACT OF A FAILURE

- Anything connected together might interact
  - Try not to connect together redundant components
- Sometimes an inoperative component can prevent another redundant component from working
  - Incorporate into the system design ways of removing, isolating, or avoiding the use of a failed component



# Q&A



# SPEAKER CONTACT INFO

**E-mail:**

**Keith.Parris@hp.com**

**Website:**

**<http://www2.openvms.org/kparris/>**



# NEXT STEPS

Lunch

Visit the EXPO to learn more about HP's High Availability products

Converge. Innovate. Transform.



OUTCOMES THAT MATTER.

