Considerations when implementing HA in DMF

DMF UG meeting
Gerald Hofer

Presented by Susheel Ghokhale

What is High Availability?

According to Wikipedia:

"High availability is a system design approach and associated service implementation that ensures a prearranged level of operational performance will be met during a contractual measurement period."

The keywords are "system" and "measurement".

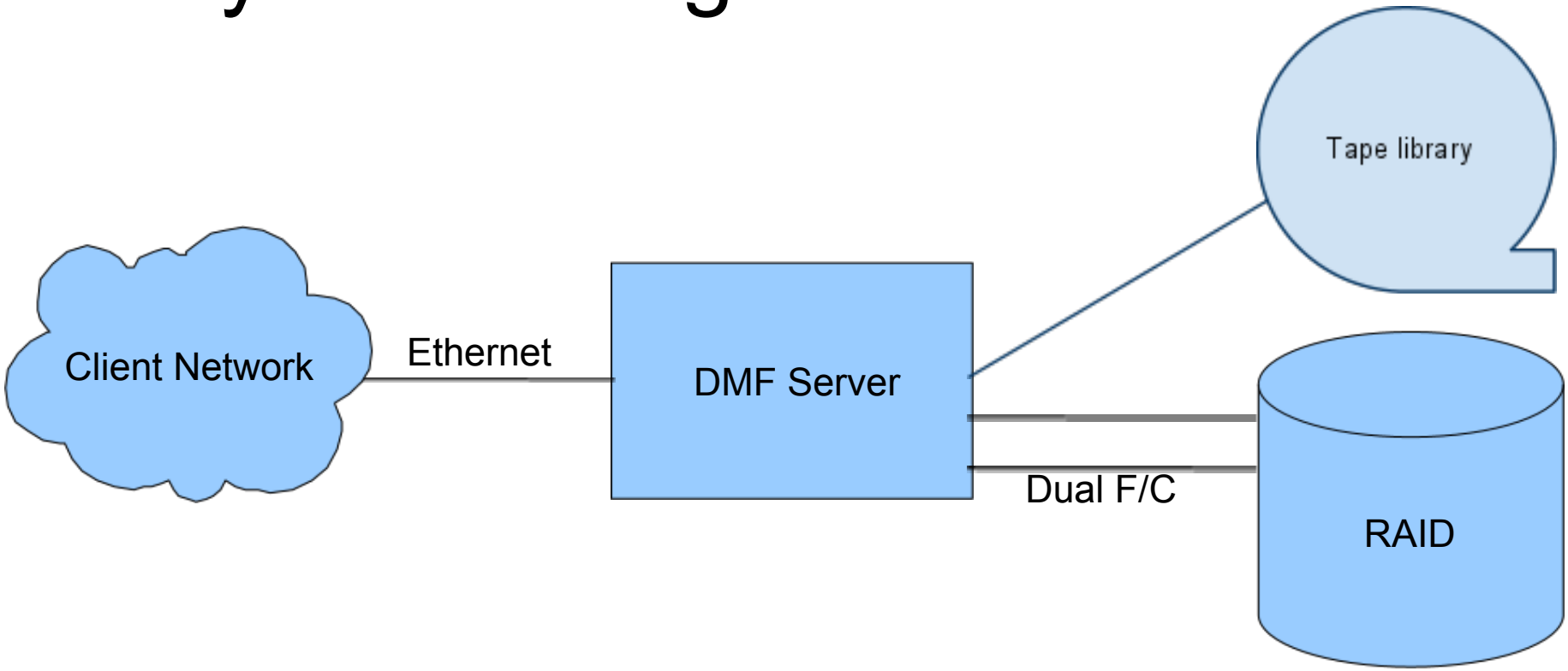
What is High Availability?

MTBF: Mean Time Between Failures
MTBR: Mean Time Between Repairs

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

- So:
  - Increase MTTF (better hardware)
  - Decrease MTTR (redundant hardware + software)

System Design Considerations



Reasonably Highly Available, Most of the Time

Redundant Hardware

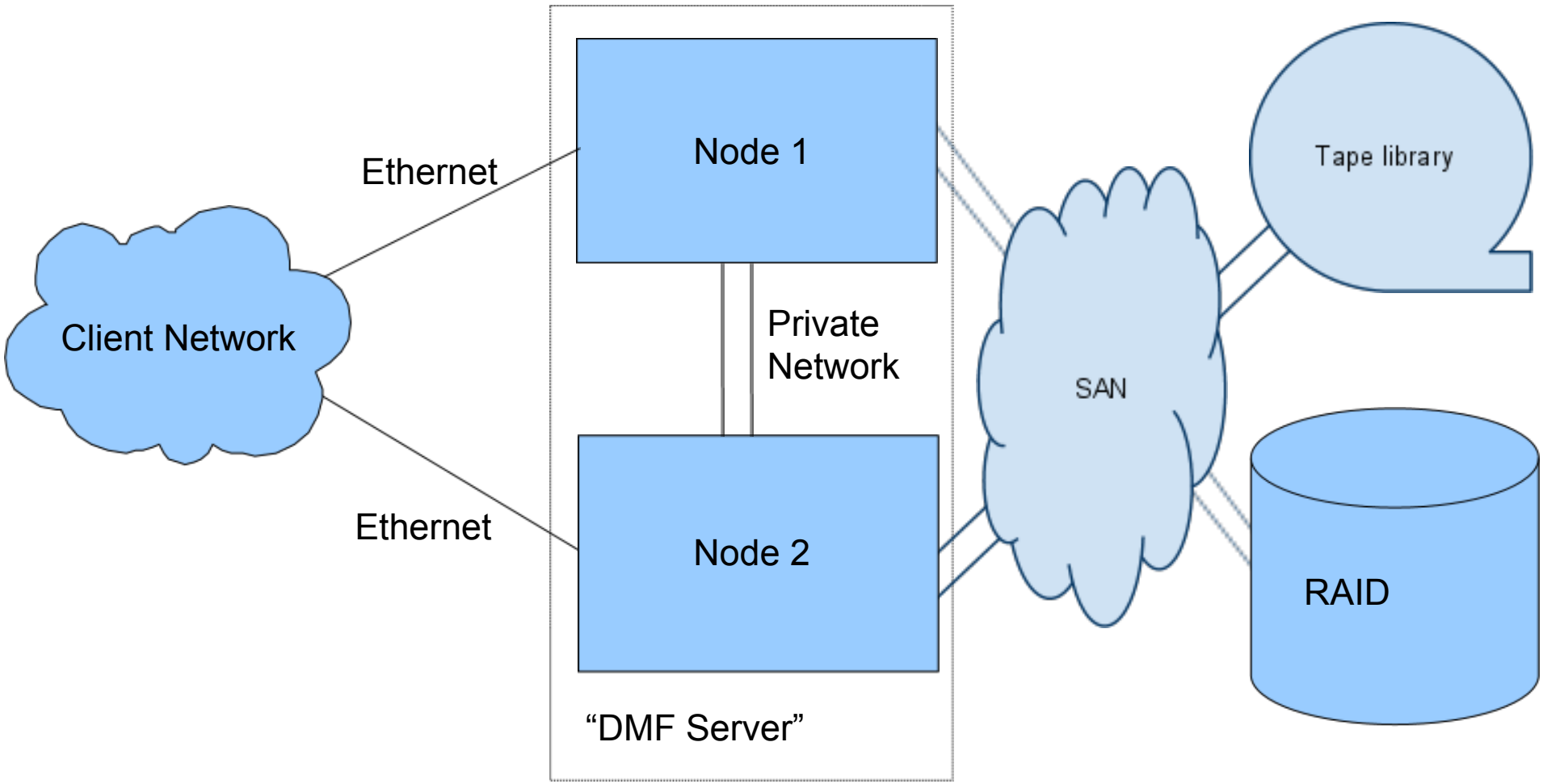
For components with lowest MTBF

- Disks
  - mirrored or Raid5/6
  - external Raid
    - is usually designed for no single point of failure
    - HBA
    - redundant RAID controllers
    - Cables
- Tape drives
- Tape libraries
- Power supplies

Single DMF system
What remains that can affect availability?

- Hardware
  - CPU
  - memory
  - backplane
- Software
  - kernel
  - applications
  - need for updates
- External factors
  - Environment
  - Power
- The human factor / Murphy
  - administrators
  - service personal

System Design Considerations



Node 2 takes over when Node 1 fails

What is High Availability?

The paradox is that adding more components and making a system more complex can decrease the system availability.

A simple single physical system with redundant hardware can potentially achieve the highest availability.

But this ignores the fact that that a single physical system needs to be brought down for patching, upgrades, testing.

Considerations

How long does it take to fix a problem?

- how long to identify the problem
- how long to get a spare part
- how long does it take to reboot

Frequency of planned outages

- updates

How much is dependent on the DMF system?

- An archive/backup system - low impact
- A DMF/NFS server for several thousand cluster cores with time sensitive applications - very high impact

History

SGI has long tradition supporting High Availability Software

- Failsafe
  - IRIX
- SGI InfiniteStorage Cluster Manager
  - SLES9 and SLES10
- SGI Heartbeat
  - SLES 10
- SLE/HAE - Novell High Availability Environment for SLES
  - SLES 11

SGI InfiniteStorage Cluster Manager

- based on Redhat Cluster Manager
- SLES9 and SLES10
- inflexible, needs shutdown for configuration changes
- still running at QUT Creative Industries as DMF/Samba server
  - lots of problems in the beginning
  - now that system has matured
  - planned to retire soon (since about a year?)

```
metal:~ # clustat
Cluster Status - QUT_CI                                    01:48:05
Cluster Quorum Incarnation #1
Shared State: Shared Raw Device Driver v1.2

  Member            Status
  ------------------ ----------
  indie             Active
  metal              Active     <-- You are here

  Service       Status   Owner (Last)     Last Transition Chk Tmout Restarts
  ------------- -------- ---------------- --------------- --- ----- --------
  QUT_CI_Fileser started  metal           20:19:03 Dec 18   0   400       0

metal:~ # uptime
  1:39am  up 65 days  5:31,  1 user,  load average: 1.10, 1.12, 1.09
```

SGI Heartbeat

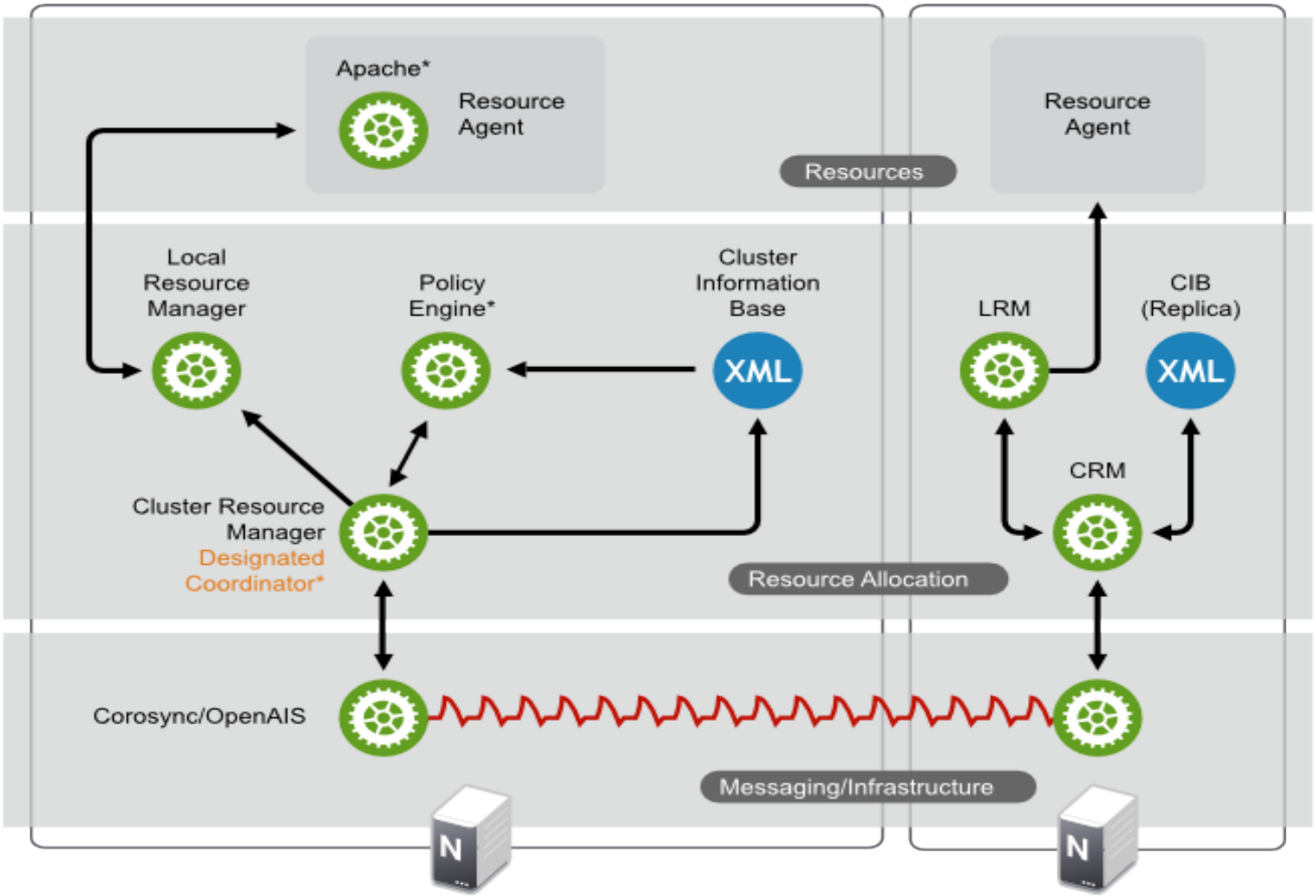
- SGI build of the Linux-HA Heartbeat package that is a product of the community High Availability Linux Project
- SGI specific modules and changes
  - cxfs, xvm, tmf, openvault, DMF, L1 and L2 controller
- SLES 10
- Heartbeat v2
- in active use at several sites in Australia
  - QUT HPC, UQ HPC, JCU, DERM
- flexible but hard to configure and administer (XML, cryptic command line)


```
<resources>
  <group id="dmfGroup">
    <primitive class="ocf" provider="sgi" type="lxvm" id="local_xvm">
      <instance_attributes id="1123e13b-69b0-4190-9638-4053acf2c23d">
        <attributes>
          <nvpair name="volnames" value="DMFback,DMFhome,DMFjournals,DMFmove,DMFspool,DMFtmp,DMFcache,archive,home,orac,pkg,scratch,static,work" id="55931f76-
c083-4a22-998b-c949746e4dcb"/>
          <nvpair name="physvols" value="DMFback,DMFback_45,DMFcache_19,DMFcache_20,DMFcache_21,DMFcache_4,DMFcache_5,DMFcache_6,DMFcache_7,DMFcache_8,DMFhome
,DMFjournals,DMFmove,DMFmove_46,DMFspool,DMFtmp,archive_34,home_0,home_1,home_2,home_3,orac_32,pkg_26,pkg_27,pkg_28,pkg_29,scratch_39,scratch_40,scratch_41,scr
atch_42,scratch_43,scratch_44,static_33,work_22,work_23,work_24,work_25,work_35A,work_36A,work_37A,work_38A,www_30" id="842b3da6-1ead-40b7-8e8b-e94a283702a3"/>
[...]
```

SLE/HAE - Novell High Availability Environment for SLES

- Novell product with support
- Based on open source components
  - Pacemaker, OpenAIS, Corosync
- SLES11SP1 and up
- SGI is adding extensions
  - cxfs, xvm, tmf, openvault, DMF, L1 and L2 controller
- Very flexible
- Much easier configuration and administration
  - powerful CLI
  - working GUI
- First installations in the region (without DMF)
  - Korea, UQ EBI mirror

Architecture



Architecture

- Resource Layer
  - Resource Agents (RA)
- Resource Allocation Layer
  - Cluster Resource Manager (CRM)
  - Cluster Information Base (CIB)
  - Policy Engine (PE)
  - Local Resource Manager (LRM)
- Messaging and Infrastructure Layer
  - Corosync, OpenAIS

Resource Layer

- Instead of starting resources during boot, the resources are started by HA
- The resources agents (RA) are usually scripts
- The most common script standard is OCF
  - Set of actions with defined exit codes
  - start, stop, monitor
- LSB scripts
  - normal init.d scripts
  - not all init.d scripts use correct exit codes

Resources example

Resource Group: haGroup
local_xvm (ocf::sgi:lxvm): Started nimbus
_dmf_home (ocf::heartbeat:Filesystem): Started nimbus
_dmf_journals (ocf::heartbeat:Filesystem): Started nimbus
_dmf_spool (ocf::heartbeat:Filesystem): Started nimbus
_HPC_home (ocf::heartbeat:Filesystem): Started nimbus
tmf (ocf::sgi:tmf): Started nimbus
dmf (ocf::sgi:dmf): Started nimbus
nfs (lsb:nfsserver): Started nimbus
ip_public (ocf::sgi:sgi-nfsserver): Started nimbus
ip_nas0 (ocf::sgi:sgi-nfsserver): Started nimbus
ip_nas1 (ocf::sgi:sgi-nfsserver): Started nimbus
vsftpd (lsb:vsftpd): Started nimbus
Mediaflux (ocf::sgi:Mediaflux_ha): Started nimbus
ip_ib1 (ocf::sgi:sgi-ib-nfsserver): Started nimbus
ip_ib2 (ocf::sgi:sgi-ib-nfsserver): Started nimbus
ip_ib0 (ocf::sgi:sgi-ib-nfsserver): Started nimbus
Clone Set: stonith-l2network-set
stonith-l2network:0 (stonith:l2network): Started stratus
stonith-l2network:1 (stonith:l2network): Started nimbus

Resource Allocation Layer

Most complex layer

- Local Resource Manager (LRM)
  - start/stop/monitor the different supported scripts
- Cluster Information Base (CIB)
  - in memory XML of configuration and status
- Designated Coordinator (DC)
  - one of the nodes in the cluster is the boss
- Policy Engine (PE)
  - If something changes in the cluster an new state is calculated based on the rules and state in the CIB
  - Only the DC can make the changes
  - The PE is running on all nodes to speed up failover
- Cluster Resource Manager (CRM)
  - binds all the components together and provides the communication path
  - serializes the access to the CIB

Messaging and Infrastructure Layer

- I am alive signals
- communication to send updates to other nodes

2 node cluster

- Most DMF HA installations are 2 nodes clusters
- DMF can only run once -> active, passive
- storage and tapes are accessible from both nodes
- Only one node is allowed to write to the file system and to tapes
- Who is the boss?
- If the HA system is not sure a resource has been stopped on a node, there is no safe way to use the resource on a different node

STONITH - shoot the other node in the head

- A reliable way to kill the other node

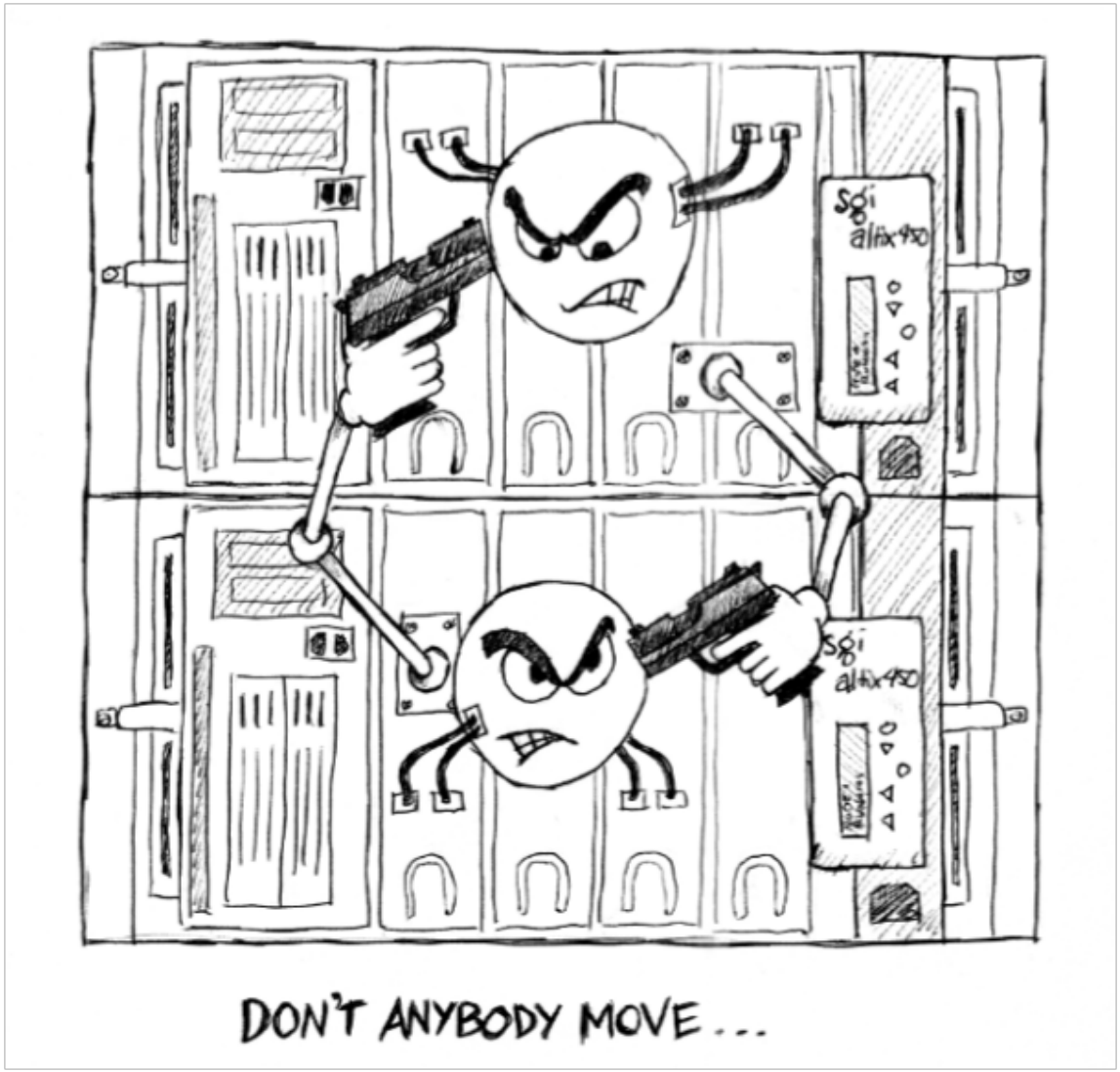
STONITH implementation

- STONITH devices are implemented as resources
- A stonithd daemon is hiding the complexity

Different physical implementations

- Remote power board
- L1/L2
- BMC

The implementation needs to make sure that when it reports the successful completion, that the targeted system has no way of running any resources any more.


sgi
altix450
sgi
altix450
DON'T ANYBODY MOVE...

Real world implementation

- Set up single DMF server
- Test it - and fix all problems
- Then convert the system to HA

Most problems with HA during the installation phase are related to the fact that the base system was not working correctly in the first place.

- System does not shut down cleanly
  - maybe only under load
- System needs user intervention after a boot
  - Switch is not using 'portfast' and the interface is not configured properly after boot

Problems - monitoring timeouts

- Resources and the other node are periodically monitored
- In case of a fault of a resource or the other node HA is initiating usually a failover, sometimes STONITH the other node
- In some cases high utilization can cause monitoring to fail without real problem
- This causes unplanned and unnecessary failover events
- Usually not a big problem for NFS (service interruption of a few minutes)
- Potentially more issue for other services (FTP, SMB, backup)

After a new installation every failover incident needs to be analysed and appropriate changes and improvement implemented. It can take a while to bed down a HA systems. Experience helps.

Problems - crash dumps

If a node is crashing the best way to diagnose the problems is to capture a crash dump.

In an HA environment the surviving node will detect the crash and issue a STONITH. Because the node was in kdb or in the middle of a crash dump this reset destroys vital debugging information.

Manual intervention might be required to capture these dumps.

Problems - syslog

Most of the information about the state of HA is logged into syslog. If a node is reset, you are usually loosing the last few syslog enties as that nodes had no time to write out the information to disk. This makes it hard to debug why a node failed.

Solution is to write the syslog also to the other node over the network.

Problems - configuration changes

Changing the configuration on the running system is possible and can be done safely.

- unmanage resources
- make changes and verify normal operation
- manage resources

But there can be some traps

- Administrators forget they are working on an HA system and restart a service without unmanaging the resources
  - if the monitoring is active that can mean a failover
- Administrators add a new file system and forget to add the mount point on the second system
  - when the service want to start it fails and it fails over to the other system again.
- The faults were not cleaned up correctly after the maintenance
  - When the system is managed again it fails over

Problems - STONITH matches

It can happen that the system gets into a state where nodes are constantly shooting each other. So every time both nodes are booted and forming a cluster, one of the nodes get shot.

This usually only happens because of configuration problems.

There are easy ways to break the loop and fix the configuration. Experience helps.

Why do you want to stay away from HA?

One of the system administration rules is (or should be):

"*Keep it simple*"

Because HA is adding more complexity, there are more ways Murphy can strike. So overall your availability can be negatively affected.

Why not use just a spare system (cold spare) and avoid HA

A cold spare system can reduce the time to get spare parts.

But there is still a considerable downtime to activate and test the cold spare system.

The biggest problem is that usually the procedures to use a cold spare are infrequently tested (they need a downtime) and changes and updates to the running system may invalidate these procedures.

Because most of the hardware is most likely in a cold spare, it makes sense to invest into HA and make sure that the spare is always usable and configured correctly.

In some cases you are better off with a good support contract.

Reasons to consider HA

Most of the time the DMF system is a central component and any outage is affecting the availability of other systems as well.

Because in HA the failing of one system is a normal operation, these fault scenarios are constantly tested. This assures that if a fault occurs that everything is working as expected. On single systems the fault scenario is rarely tested and a lot of times uncovers problems at the most inconvenient time.

In HA systems both systems are already booted and the standby system is ready to steal volumes and start services. A failover event is much faster then a reboot of a system.

Updates and upgrades can be done online with minimal interruption.

References

Novell SLE/HA
http://www.novell.com/documentation/sle_ha/pdfdoc/book_sleha/book_sleha.pdf
http://www.clusterlabs.org/wiki/Documentation
http://ourobengr.com/high-availability-in-37-easy-steps.odp