Clusters and High Availability

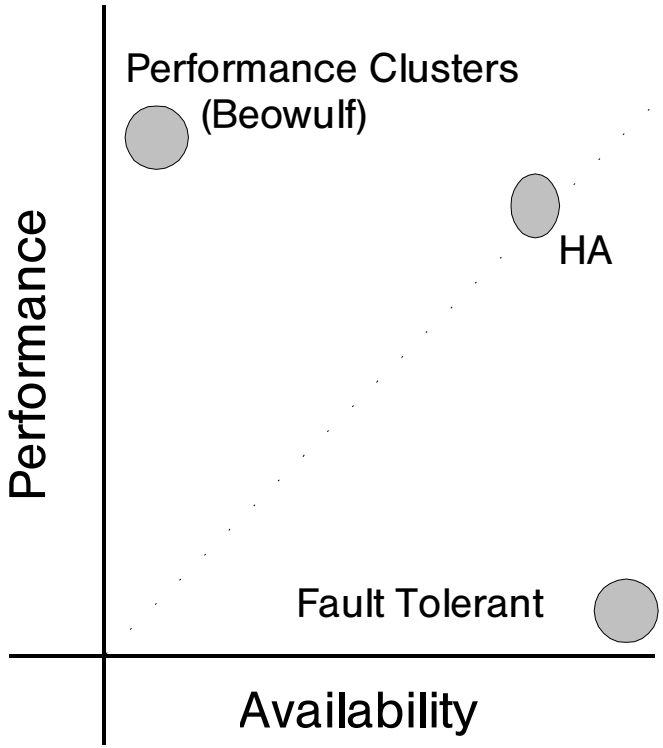
James Bottomley
SteelEye Technology

Introduction to High Availability using commodity hardware clusters

Types of Cluster

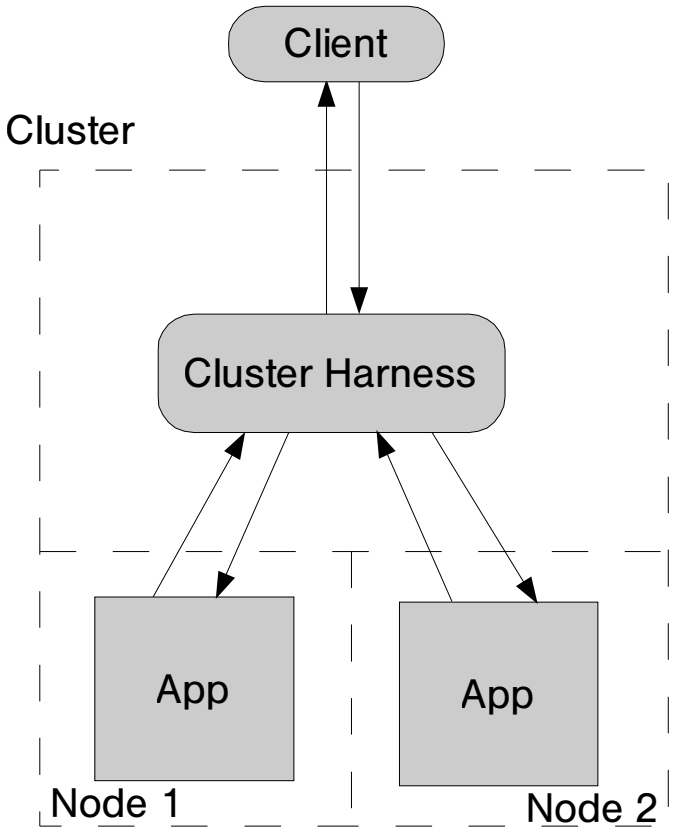
- MPP - Compute clusters (Beowulf)
- FT - Full Fault Tolerance (Isis, Stratus)
  - Recovery undetectable, no transaction loss
- HA - High Availability (LifeKeeper, Linuxha)
  - Detectable recovery (seconds). Uncommitted transactions may be lost
- DR - Disaster Recovery
  - Committed transactions may be lost
- LB - Load Balancing

Availability Chart



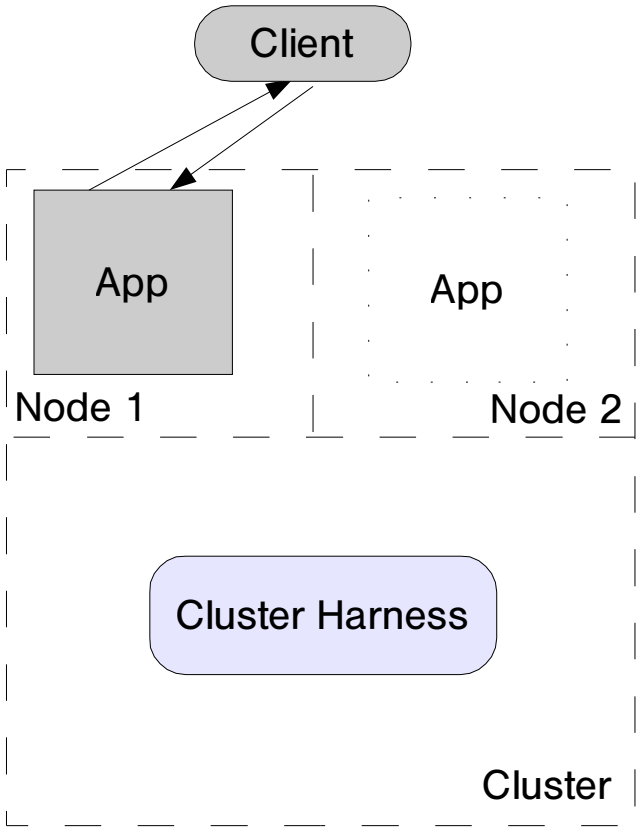
- Performance Clusters cannot tolerate instantaneous failure
- HA can provie availability with failure with virtually no performance degradation
- FT sacrifice performance for avaliability

Fault Tolerance



- Multiple application copies.
- Harness splits input and monitors output.
- If >2 copies, may identify failing output.
- Applications must use deterministic transactions.

High Availability



- Single copy of the application.
- Recover only when fault detected.
- Recovery time depends on application and resources.

Failure Modes

- Local
  - Application fails but cluster node still operates.
  - May recover application locally or on a different node.
  - Recovery Mediated by node where failure occurs.
- Global
  - Entire node fails (or hangs).
  - Recovery mediated by cluster.

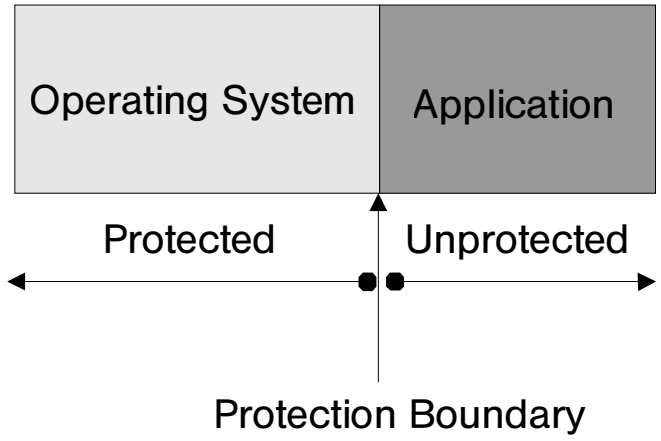
Failure Detection

- Local: Must have customised watcher that monitors the services the application provides (simply checking the process isn‘t enough)
- Global: Every node watches every other via multiple distinct communication paths. When all paths fail, the node at the other end is assumed to be dead.
- Must prevent spurious Global failure detection caused by comm path failure.

Pathological Failures

- These are failure modes induced by the setup:
  - Single Points of Failure (SPOFs) outside the control of the cluster harness.
  - Total or partial failure of the cluster communication paths causing one or more nodes to appear dead.
  - Complete or partial system hang.
- Eliminate SPOFs by careful hardware design provisioning and deployment.
- Cluster software must prevent inappropriate response to comm path failures.

Pathological Application Failures



- HA moves protection into app by monitoring
- If app failure due to data input, no amount of restarts will fix the problem
- Fortunately, most apps well behaved.

Recovery Requirements

- Application itself must be capable of recovering from a crash.
- All resources that may be used by the application (open files, database connections, IP addresses etc.) must be available.
- If all resources are available, the application may be recovered on a different node.
- Applications are resource aware not location aware.

Performing Recovery

- If the Recovery Requirements are satisfied, most of the work is done by the application
- The HA software only has to provide the resources and then start up the application.
- Clients of the application see an interruption in service while all this is going on.
- The level of interruption depends on how the application is coded to handle crashes.

Providing IP resources

- Moving an IP address from one machine to another is easy (use ip aliases).
- Getting other machines to see the switch is harder because of ARP caching.
  - Any machine on the network may respond to an ARP request if the mapping is in its cache.
  - Gratuitous ARP (GARP) is one method of forcing the switch to be seen (used by linux-ha).
  - ARP cache flushing is another (used by LifeKeeper).

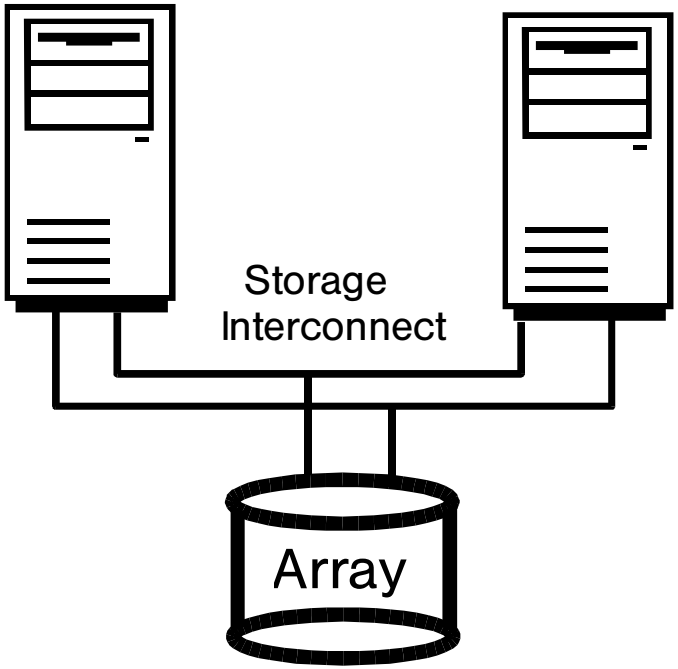
Providing Storage Resources

- Need either physical access to the storage from all nodes in the cluster (shared SCSI, SAN etc.) - Expensive
- or replicated copies - Cheap but a network bandwidth hog (also less reliable).
- Pathological failure modes can produce data corruption unless you have I/O Fencing.
  - STONITH devices
  - Watchdog timers
  - SCSI Reservations (Shared storage only)

Replication

- For HA, replication must be synchronous
  - Block only acknowledged as committed when it reaches the storage on both primary and secondary.
  - Adversely affects latency.
- If connection breaks, must resynchronise entire volume! Mitigate with:
  - Transaction Log (unbound but doesn‘t corrupt secondary on replay).
  - Intent Log (known size but corrupts secondary during log replay).

Shared Storage

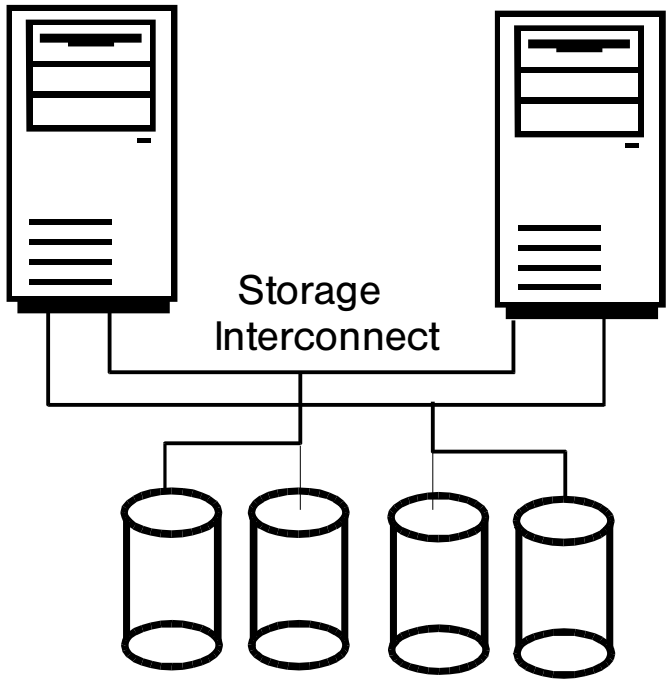


- External RAID array expensive
- Eliminating Single Points of Failure (dual loop) even more costly
- However, often a choice for large dataset (enterprise) environments.

SANs

- Storage Area Networks
  - Fibre Channel Based
  - Essentially bigger and better shared SCSI.
- SAN Problems inherent in Linux but impact Clusters:
  - Lack of decent device identification and management infrastructure (but see LVM, EVM)
  - Lack of large number of device support (possibly change in 2.5).
  - Need a useful device node naming scheme.

Shared Host Based Raid



- Raid Controller is inside each node
- Slightly Cheaper Alternative to Shared Storage
- RAID cards must be cluster aware
- I/O fencing problems even more acute (destroy RAID)

STONITH

- Shoot The Other Node In The Head.
- Implemented as a serial line controlled power supply for the whole cluster.
  - Obviously a SPOF.
- When Node A detects failure on Node B it turns off Node B‘s power.
  - Active protection.
  - Races mediated by Stonith device.
  - No more effective than an additional comm path for pathological comm path failure.

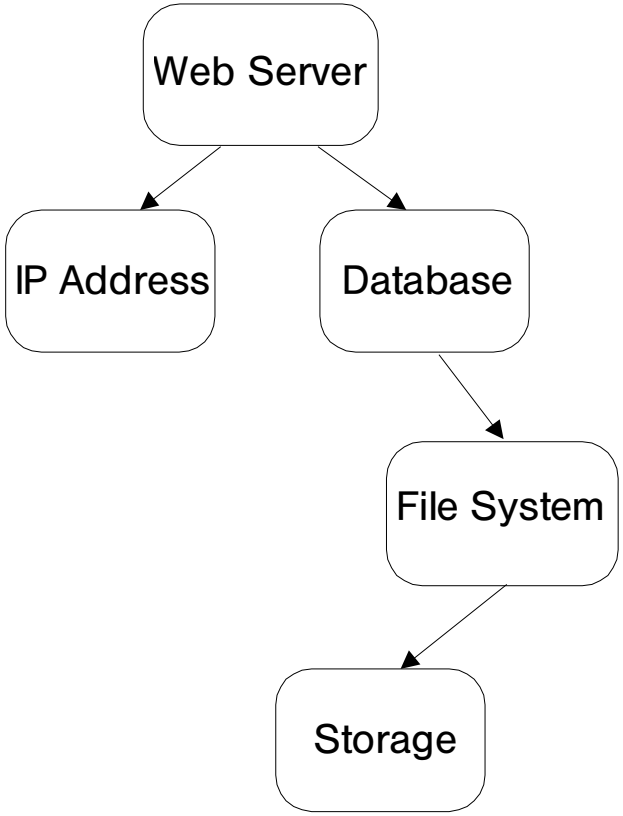
Watchdog Timers

- Must be prodded by system periodically or they power off the machine
- Primarily cure I/O fencing problems caused by system hangs.
- Do not cure problems caused by pathological comm path failure
  - Can mitigate this by making the storage access a communication path (disc based mailbox).

SCSI Reservations

- Most reliable: Access mediated by the device itself.
- Unfortunately, not supported by all devices
  - and even supporting devices may have quirky implementations.
- Not implemented in the vanilla Linux Kernel
  - Most OS distributions (RedHat, SuSE etc.) have support in their kernels.
- Reservations may be used to arbitrate a total comm path failure situation.

Recovery By Hierarchy



- Applications may depend on other Applications and Resources
- Must perform an ordered recovery
- Hierarchical divisions also make monitoring easier

HA versus Disaster Recovery

- Difference is in committed transactions:
  - HA must not lose committed transactions
  - Disaster Recovery (DR) may lose a pre-specified number of committed transactions.
- Very difficult to automate DR unless you are sure you need take no corrective action for the lost transactions.
  - Bank‘s cash machine transaction? Probably not
  - Non revenue record update? Possibly

What is a Transaction?

- Almost any action for which you have an agreement followed by an action.
  - e.g. Buing an item is a three stage transaction: Agree on price, pay price, take item.
- In computer terms, a transaction is defined as an abstraction of an atomic and reliable execution sequence.
- *Idempotent* transactions are ones which may be repeated without affecting anything.

Disaster Recovery Criteria

- DR protection done by geographically dispersed data replication
  - Large distances make lowering latency almost unviably expensive for synchronous replication
  - Must replicate asynchronously to avoid latency but must preserve data ordering so replica is always an Out Of Date copy of Primary.
  - Limit transaction loss by limiting the amount of sent but unacknowledged data blocks between the primary and the secondary.
  - Tuning this ”window” allows a Bandwidth for Latency swap.

Transmission Interruption

- If transmission is interrupted, mirror breaks.
  - When transmission restored, must avoid resending the entire primary data set.
- Do this by keeping a log of Sent but Uncommitted transactions.
  - Only need to replay this log to bring replica into sync with primary
- Two types of logging
  - Transaction.
  - Intent.

Transaction Log

- Keep an ordered local log of all data blocks that would have been sent to the replica.
- Can thus replay log in order, so replica is out of date but never corrupt.
- Since data contents are logged, log space requirements are large.
- Log grows without bound and could overflow available space for long transmission line interruptions.

Intent Logging

- Instead of logging actual data, just keep a record of where the data has changed between primary and replica.
- On replay, just send changed blocks.
- Ordering not stored in log, thus replica is corrupt while log replay is in progress.
- Log can be a bitmap covering volume and is a known non-increasing size.

Conclusions

- Well designed applications recover themselves.
- HA software must look after resources and monitoring.
- HA software must plan for and cope with all pathological failures.
- Replication can be used for HA or DR but using different characteristics.