Resource Driven Clusters

James Bottomley
SteelEye Technology

OCF Workshop 20 January 2003

Introduction

- Quorate clusters are centrally controlled
  - Analagous to single CPU controlled by 1 clock
  - Cluster must form first before actions taken
  - Cluster directs all actions based on its controlling view of the cluster membership
  - Membership must be well defined
  - Actions generally agreed to by all cluster members (single cluster view)
  - Only a single cluster entity may exist at one time

Introduction (2)

- Resource driven clusters are more chaoti
  - Act like Asynchronous CPU designs (actions trickle through instead of being co-ordinated centrally)
  - There is no central controlling cluster
  - Actions controlled for a given resource by cluster member who ”owns” the resource
  - Other member acquiescence to actions by owning node not required

Introduction (3)

- Resource driver clusters (continued)
  - No central cluster means no monotonic instance numbers
  - Cluster may form with partial communications
  - Multiple resources => multiple owning nodes each of which may take an action simultaneously
  - Multiple independent sub-clusters may form

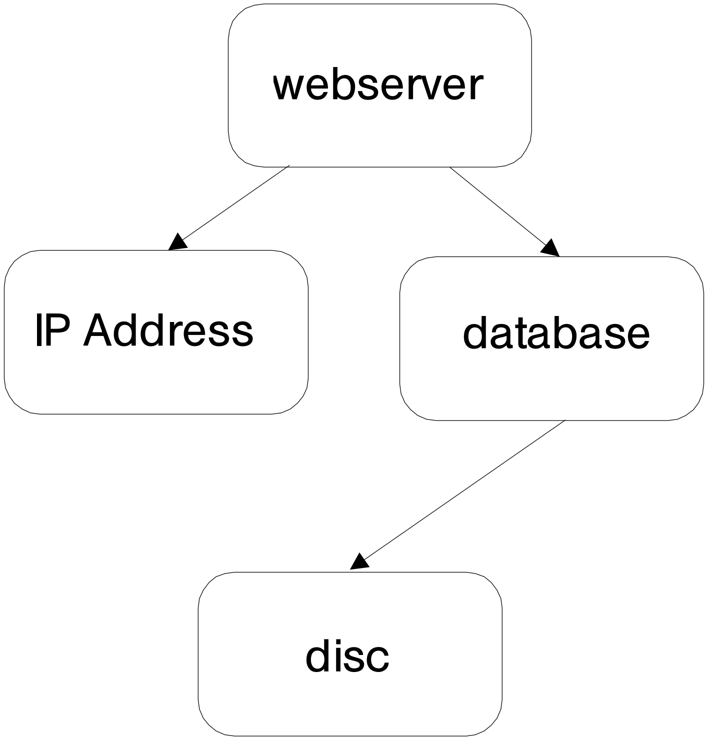
Why?

- Easier to design and build (no central control layer need be constructed)
  - Simplicity is desirable in HA (less to go wrong)
- Better scaling properties (in large clusters with large numbers of resources)
- Better disaster survivability (formation of multiple sub clusters usually gives better recovery characteristics)

Why Not?

- Harder to analyse.
  - Chaotic behaviour makes provability difficult.
  - Disliked by acadaemia for this reason.
- Multi-threaded failover characteristics may cause OS resource problems.
- Single cluster view hard to obtain
  - makes administration difficult
- Definitely not like the good old VAX

Elements of Resource Driven Clusters



- Resources comprise Hierarchies
- Hierarchies are fundamental units
- At least one resource of a hierarchy must be ownable.
- arrows represent dependencies

Resource Ownership Properties

- Classes of resources are ownable
- Ownability implies two properties
  - May I own (i.e. test of ownership)
  - Take ownership (must be exclusive)
- Disk resources implement ownability usually with reservations
- May also introduce ownership carrying resources (similar to a quorum disc)

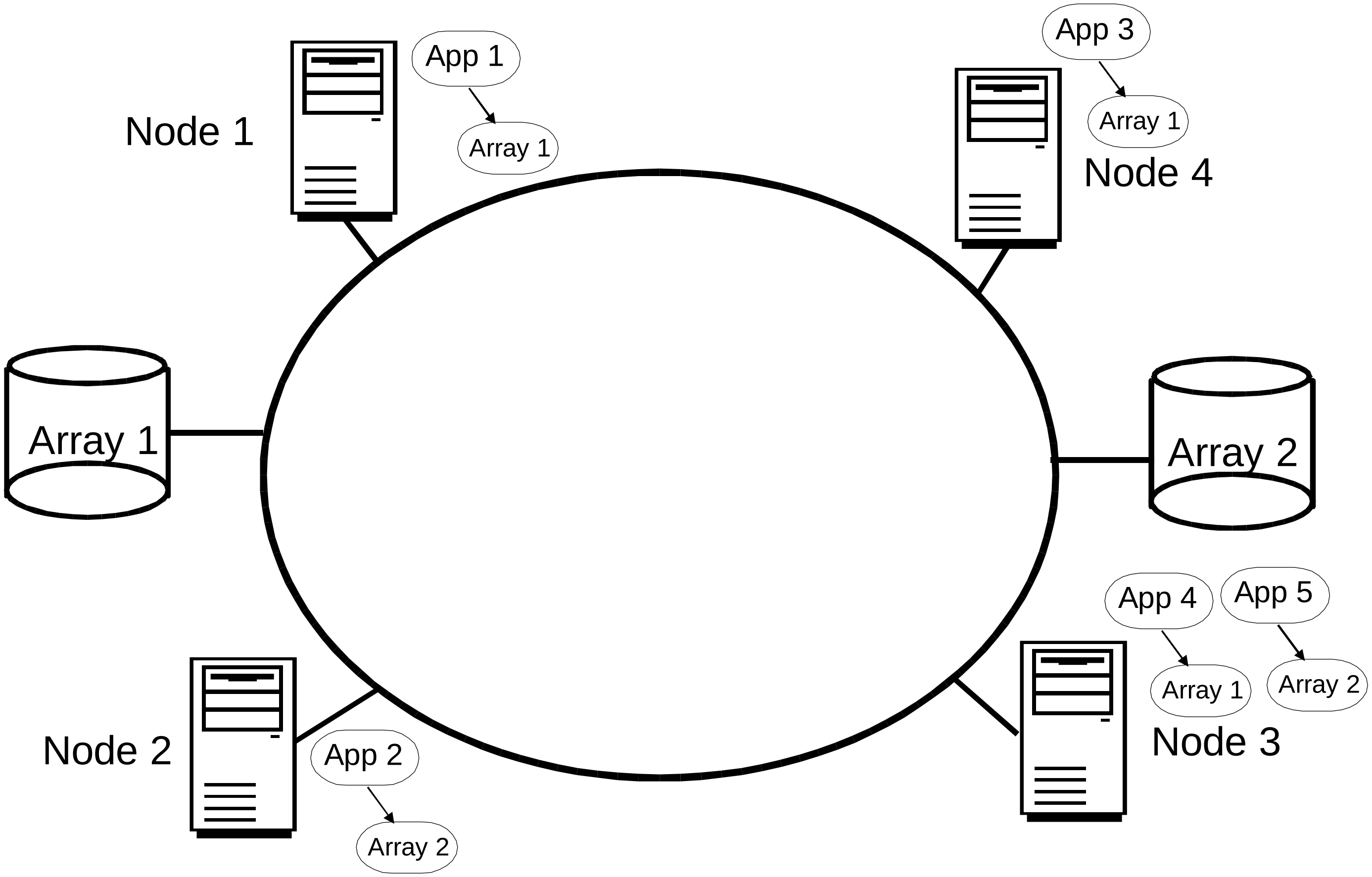
SCSI Reservations

- Tailor made for resource ownership
- Reservation will enforce exclusive access to the owning node. Another node may not accidentally or maliciously interfere with the data
- Ownership is at the disc level, not the partition level (multiple partitions move together)
- Reservations can cause OS problems (i.e. can‘t read the partition table)

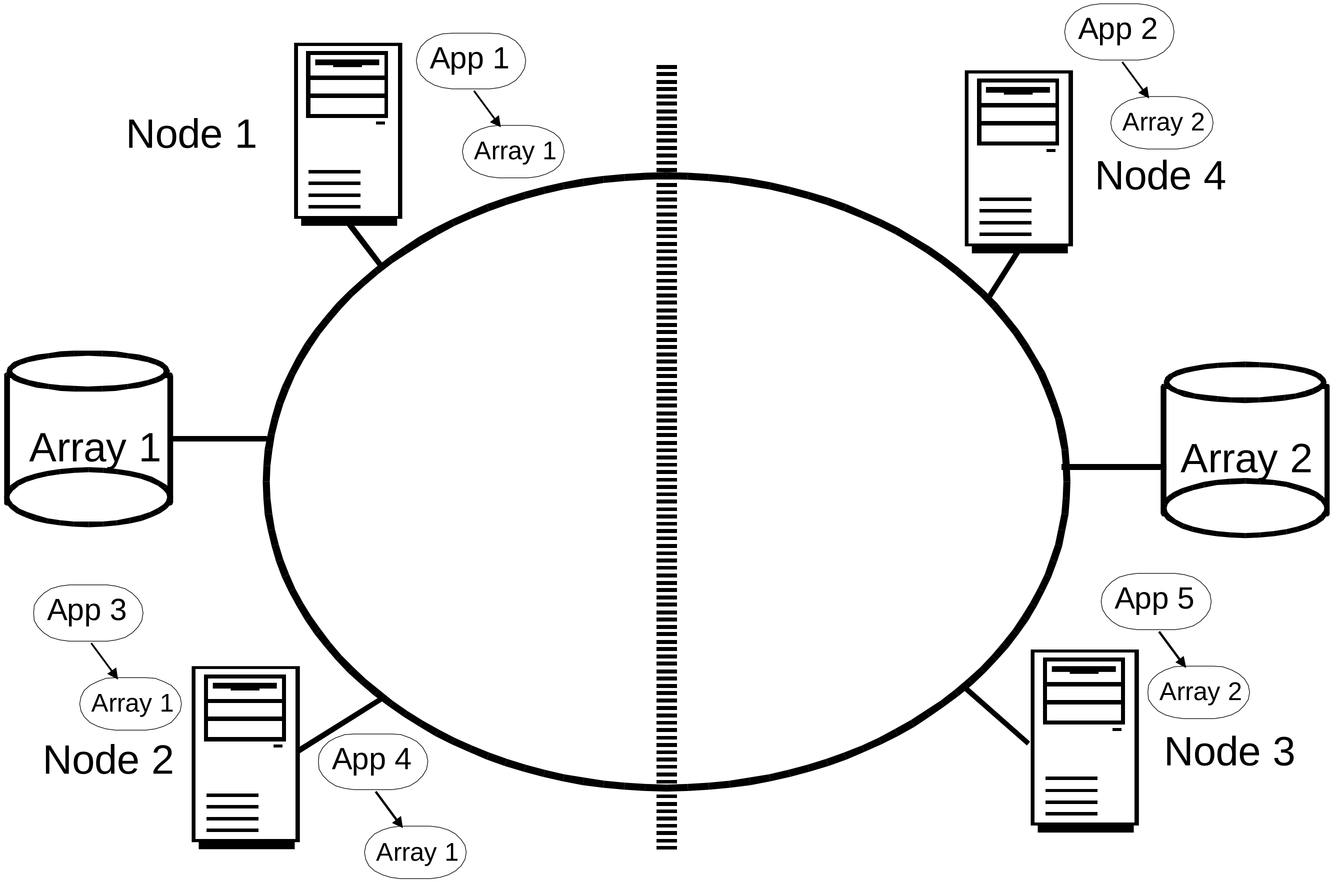
Hierarchy Ownership

- Nodes own the hierarchy
- To own a hierarchy, a node must own all of its ownable resources.
- To prevent ownership deadlock, hierarchies need a deterministic ownership acquisition ordering.
- As soon as a node owns a hierarchy, it may proceed to recover that hierarchy regardless of what is going on in the cluster.

Cluster Partition Illustration



Cluster Partition Illustration (2)



Utility Functions

- Problem: Recovery may proceed but resulting hierarchy may not be useful
  - Webserver recovers but cannot see router => no external visibility for website
- In a partition, other sub-cluster may be more useful for recovering the resource
- Therefore, construct a utility measure of the hierarchy by summing the individual utilities of the resource
  - utility is often user or application defined

Utility Functions (2)

- Utility isn‘t all or nothing, like ownership
- Utility is often computed as a ”score” out of a fixed (but small) number
- The utility score is used as a starting point for the backoff algorithm
- Backoff is the time between having the hierarchy ready for recovery and beginning the ownership acquisition
- Feeding utility into backoff gives time for a more useful node to recover first.

Utility Functions (3)

- Utility has no ”veto”
  - A completely useless hierarchy will still eventually recover (after waiting for more potentially useful ones)
  - This is by design (customers don‘t like losing heirarchies just because they‘re useless)
- Utility is probabalistic, not deterministic.
  - Circumstances may still conspire to recover a useless hierarchy over a useful one.

Limitations

- What happens if hierarchy has no ownable resources?
  - e.g. storage is replicated not shared.
  - Application requires no storage.
- Must introduce ownability into the hierarchy
  - May use spurious ownable resource (similar to a quorum disc)
  - May use other ownership tricks, like STONITH
- In no case will the model be worse than that of a quorate cluster.

Formulating Cluster View

- Difficult to obtain unambiguously
- Each node only has partial information (enough for itself to operate)
- Forming a cluster view for administration is a problem
  - Need to collate cluster view, taking input from each node
  - agent that forms cluster view needs contact with each node (may need superset of cluster communication paths)

Conclusions

- Resource driven clusters are significantly different from Quorate ones
  - Severely limits APIs covering both
  - cannot assume that the cluster is all.
- Resource driven clusters have greater flexibility and greater complexit
- Richer sequence of recovery scenarios (c.f. utility functions)