Bright Cluster Manager Failover

Cosmin Dumitru
Niek Timmers

OS3 SNE
University of Amsterdam

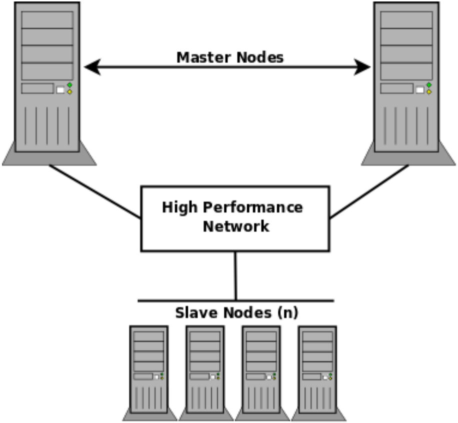
February 3, 2010

Table of Contents

- High Performance Computing
- Bright Cluster Manager
- Research Question
- Failover Mechanism
- Failover Mechanism Issues
- Failover Mechanism Proposals
- Conclusions
- Questions

High Performance Computing - HPC

- advanced computation problems
- clusters
- scientific research
- business world
- complex design



Bright Cluster Manager

Solution for installing, monitoring, managing and using clusters.

Design Goals

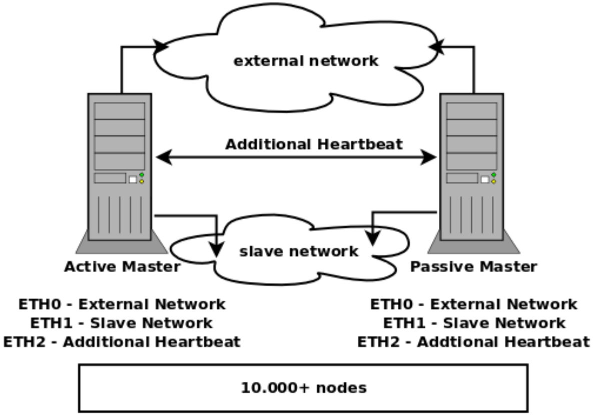
- Easy
- Scalable
- Complete
- High Availability

RP1 - Research Question

Is the failover mechanism implemented in *Bright Cluster Manager* working as intended and can it be improved?

Failover Mechanism

- 2 Nodes
- Heartbeats
- Quorum
- Fencing
- STONITH
- Shared Resource



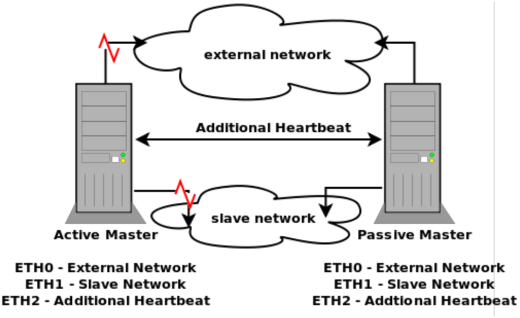
Failover Mechanism Issues

- Additional Heartbeat Link
- Local Disk Failure
- Service Monitoring
- NTP Configuration
- Failover Toggle Switch

Additional Heartbeat Link

Context

- Additional heartbeat link via additional network
- Regular ICMP ECHO_REQUEST



Additional Heartbeat Link

Problem

- Failover not initiated if default heartbeats fail
- Heartbeat via additional network is a constraint
- The slave network link is not seen as a critical cluster resource

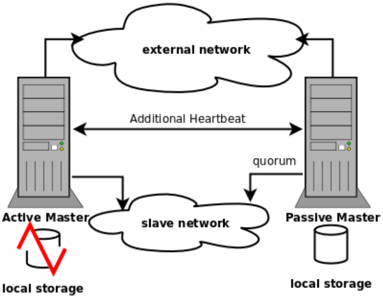
Solution

- Connect heartbeat NIC's to slave network
- Connect heartbeat NIC's to different switches
- Do not use the additional heartbeat

Local Disk Failure

Context

- Local disk failure of the primary master
- Disk is not readable (i.e. not mountable)



Local Disk Failure

Problem

- Heartbeats do not fail
- No failover initiated

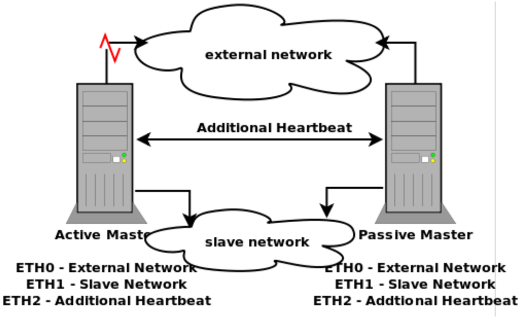
Solution

- Periodic checking of local file system
- Usage of system calls

External Network

Context

- One NIC of the head nodes is connected to the Internet
- Users connect via the external network to the cluster



External Network

Problem

- Cluster becomes unusable
- No failover is initiated

Solution

- Monitor external network interfaces
- Redundant network interface cards

Service Monitoring

Context

- Essential services are monitored
- Crashed services are restarted

Problem

- Only restart action implemented
- No action (failover) on continuous crashes

Solution

- Implement continuous failure monitoring
- Initiate failover when crash threshold is hit

NTP Configuration

Context

- Head nodes run NTP service
- Used by slaves for reliable time

Problem

- Head nodes only include external sources
- No external network means no reliable time

Solution

- Include other head node as last reliable time source
- Even with no accurate time the nodes are not affected

Failover Mechanism Toggle

Context

- Failover mechanism is triggered whenever all heartbeats die

Problem

- No simple way of shutting down the failover mechanism
- Failover system will turn active machine off (STONITH)

Solution

- Simple on/off toggle button in the GUI for maintenance

Proposals

- Active-Active
- Large quorums
  - Optimized voting sequence
  - Resource groups

Active - Active

Approach 1

- Split the cluster between head nodes
- Assign partitions to each head node
- Control is divided between the two master nodes
- Job scheduling software needs to be modified

Active - Active

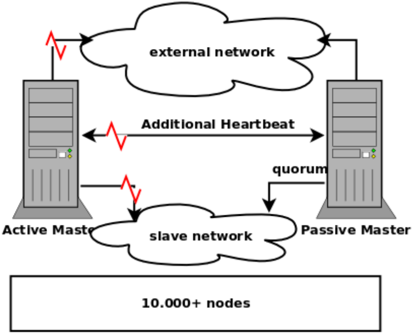
Approach 2

- Replicated services
- Virtual synchrony - intercept calls and distribute them to the group
  - Totally ordered
  - Reliably delivered
- Delay introduced by the group communication
- Complex wrapper over existing job scheduling software
- JOSHUA research proof-of-concept (Engelmann et. al 2006)

Quorum

Context

- Quorum votes are send via unicast (sequential)
- Performs perfectly in small/medium clusters



Large Quorums

Problem

- Slow quorum time caused by delay
- The quorum timeout needs to be fine tuned depending on the cluster size

Solutions

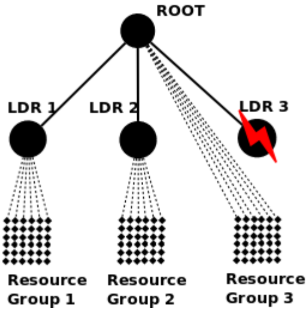
- Optimize sequence
- Resource groups

Load Based Quorum Sequence

- Change the quorum order dynamically
- Metric based on load
- Optimistic voting

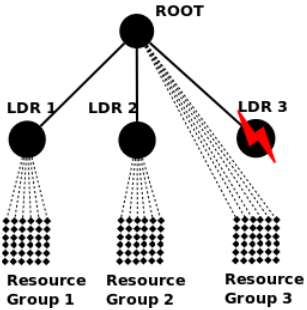
Resource Groups

- Partitioning the cluster (e.g. in a tree)
- Decrease quorum initialization time



Resource Groups(2)

- Each group has a leader (can be statically defined)
  - Monitors group nodes
  - Starts quorum inside its group
- New quorum procedure
  1. Passive Master sends leaders a command to start the quorum
  2. Leaders start quorum inside group
  3. Slaves send vote to master nodes
  4. Master node waits a small amount of time to get a majority
  5. Master node continues regular quorum procedure on slave nodes that didn't send votes in yet
- Quorum time is decreased



Conclusions

- Logical design can be improved
- Current system monitoring can be improved
- Additional software testing is needed
- Quorum design can be optimized for large clusters
- Proposals add complexity. Difficult to design and implement.

Questions