Reliable Locking for Clustered NFS Servers

Brad Boyer and Paul Massiglia
Symantec Corporation

Agenda

1. Background
   1. The environment
   2. NFS lock management
2. Problem description
3. Existing solutions
4. Our solution
5. Future work

Background

Veritas Cluster Server (VCS)
by Symantec

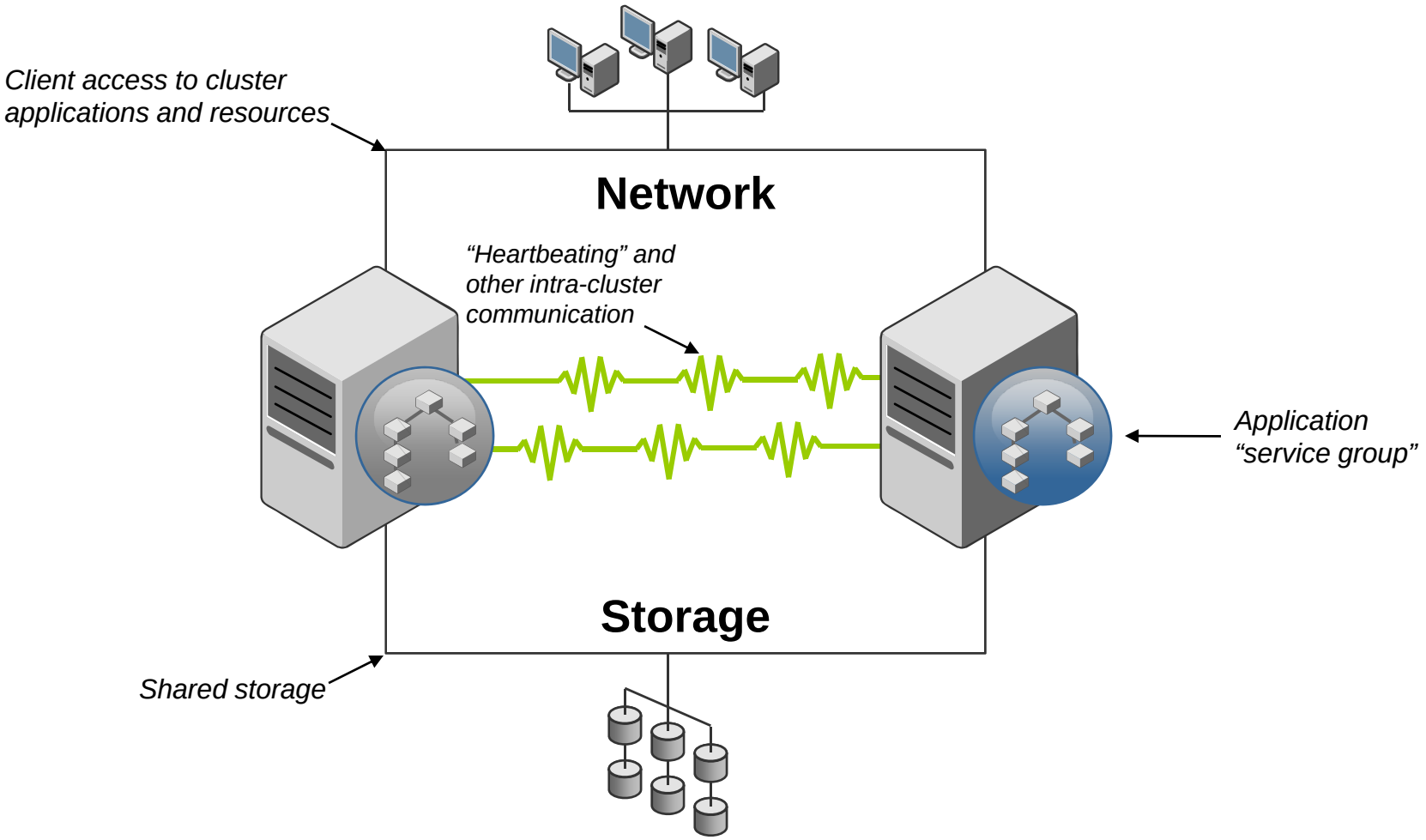
Paul Massiglia

VCS

- Loosely-coupled shared-data clustering of *NIX servers
  - Intel/AMD and enterprise UNIX computing platforms
  - Fibre Channel and iSCSI shared storage
- Parallel (scaling) applications
  - Instances run on each cluster node
  - Access data on shared storage devices
- High-availability applications
  - One instance runs on a designated primary node
  - Automatically fails over to an alternate node if primary fails

VCS cluster

Physical configuration



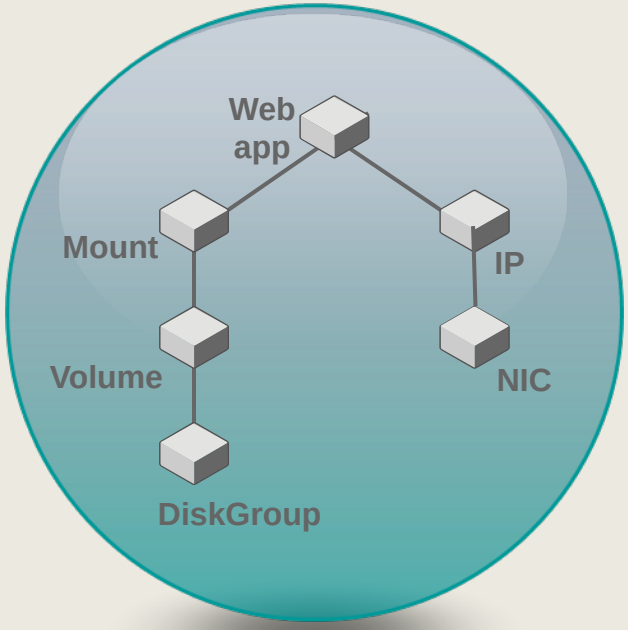
VCS Service Groups

The construct used by VCS to manage application services

consists of…

- Resources
  Components required to provide the service
- Dependencies
  Relationships between resources and with other service groups
- Attributes
  Parameters and startup/failure behaviors

Example: Service Group WebSG



Two key applications of VCS

- Cluster volume manager (CVM)
  - Unified cluster-wide view of server-based virtual volumes
    - Mirroring, striping, multi-pathing, snapshots, thin provisioning…
  - Simultaneous access by all nodes
  - Synchronized state changes

Robust underpinning for…

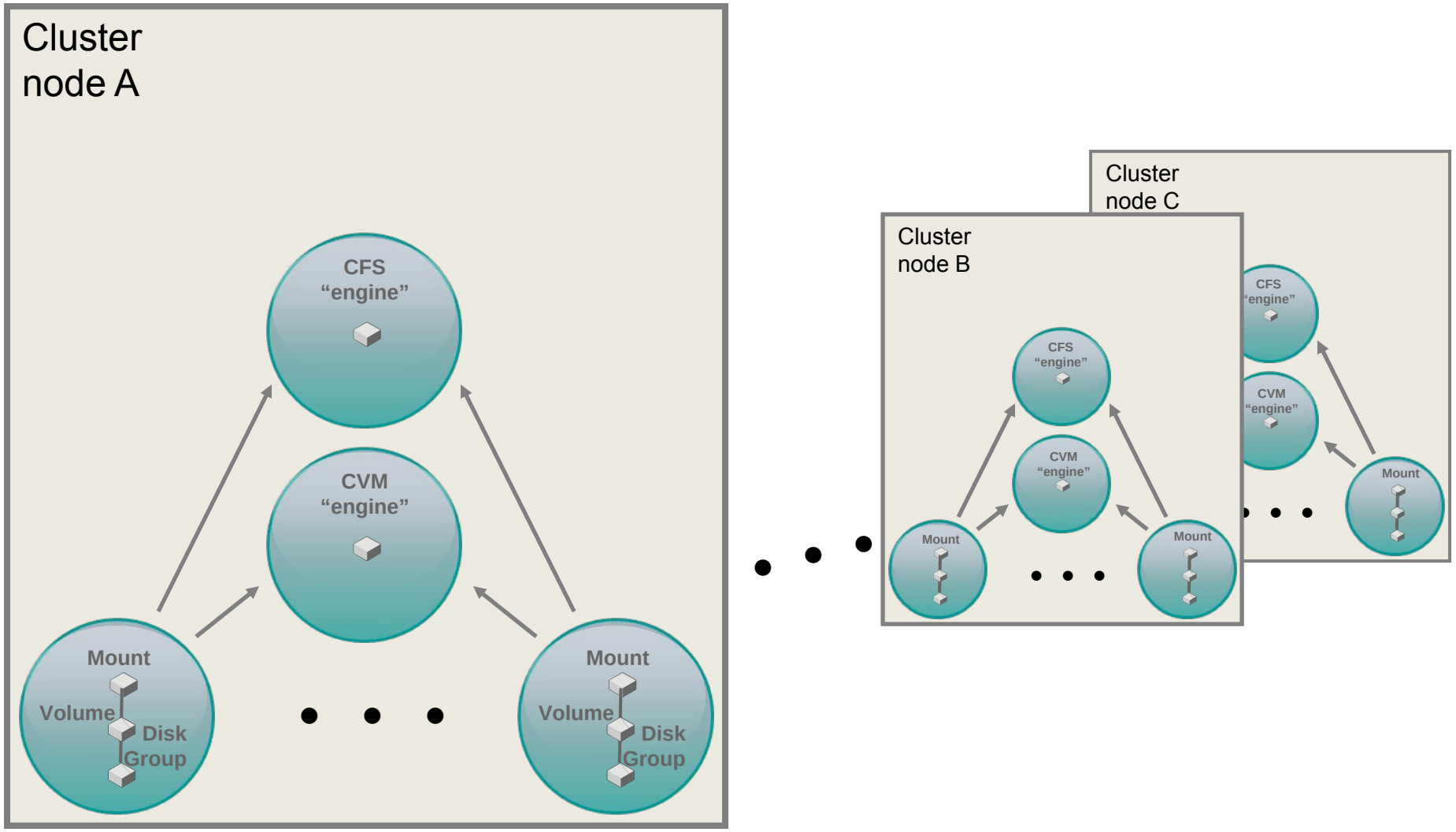
- Cluster file system (CFS)
  - Simultaneous shared access to file systems from all nodes
    - Dynamic storage tiering, snapshots and clones,…
  - Concurrent file access for parallel applications
  - Fast failover for high availability applications

Obvious VCS/CVM/CFS application

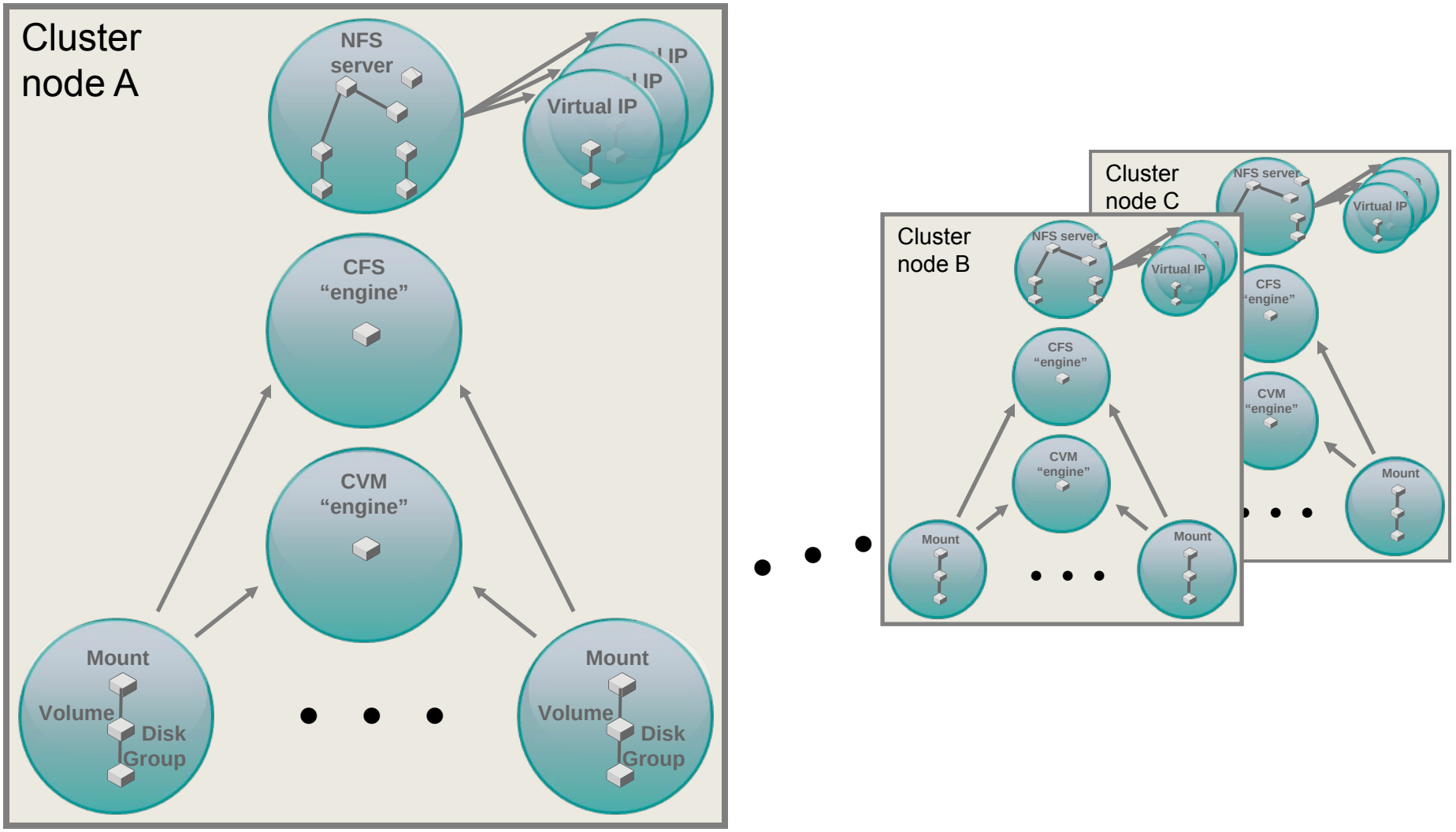
- Scale NFS services beyond single-server capacity
- Feasibility
  - Shared volumes
  - Shared files
  - Open-source NFS server component
- Need
  - Strong demand from Storage Foundation users
  - Demonstrated by success of Scalable File Server (2007-present)

Cluster file systems

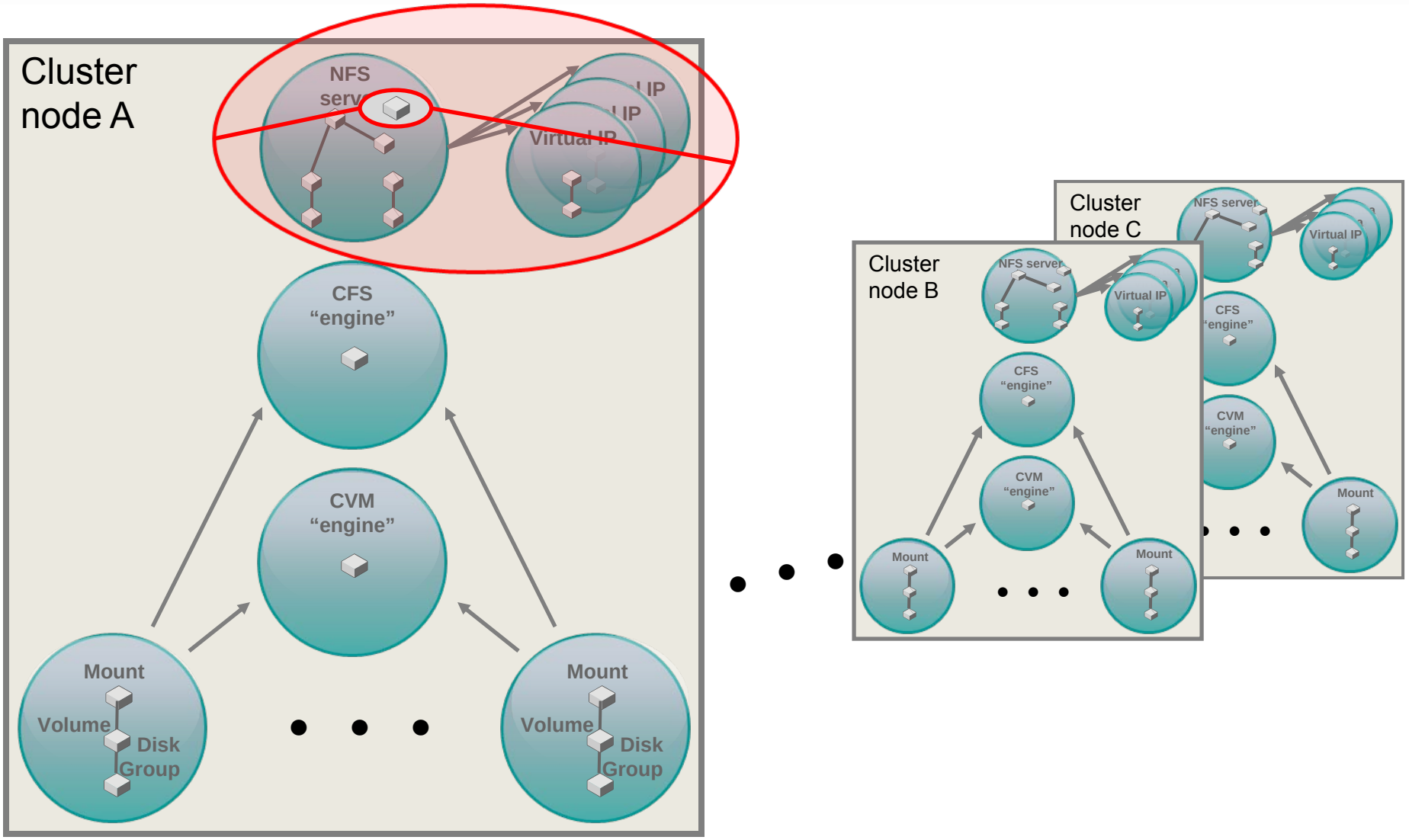
as VCS services



The short leap to serving NFS



The short leap to serving NFS



Background

Resource locking in an NFS environment

Brad Boyer

NFS Lock Basics

- NLM (Network Lock Manager) protocol
  - Lock/Unlock/Test requests
  - Used to manage lock state information on server
- NSM (Network State Monitor) protocol
  - Monitor/Notify requests
  - Used to trigger actions when a system restarts

NFS Lock Recovery

- System reboots require lock recovery
  - Both server and client must monitor each other
  - Each side has explicit steps to take after detecting reboots

- Recovery steps for server restart
  - Lock server starts in "grace mode" and rejects normal requests
  - Clients resubmit lock requests with reclaim flag
  - The "grace mode" lasts some amount of time (90s is common)
  - Lock server resumes normal request processing

Standard host monitoring

- Integration of lock server with status monitor
  - For any client with a lock granted, ensure the client is monitored
  - Discontinue monitoring client known to no longer hold any locks
  - Release locks held by client on client reboot notification
- Integration of NFS client with status monitor
  - For any server handling locks, ensure the server is monitored
  - Reclaim locks granted by server on server reboot notification
- List of monitored systems is saved in persistent storage

Problem Description

Doing it all in a cluster environment

Desired features in a clustered NFS service

- Highly-available NFS service
- Redundancy
- Distributed NFS load for performance
- Scale server and storage capacity independently
- Use commodity hardware and existing software components
- Support Dynamic Multi-Pathing (DMP)
- No client side changes

Issues with multiple active servers

- Lock requests must be guaranteed coherent results
  - The state of a lock must appear the same from each cluster node
  - This is logically tied to the common view of storage
- Lock servers must appear to all be valid or not valid
  - A lock server normally releases all locks on exit
  - Race conditions must be prevented while any server is restarting

Existing Solutions

Use bigger hardware / appliance

- Simple solution is using more powerful systems
  - May have high cost for very large systems
  - Does not take advantage of any possible partitioning of load
- Appliance model allows purpose-built code, but does not allow general-purpose use
  - Feature set is generally fixed by the vendor
  - End result may be more dependent on one vendor than desired

Single lock master

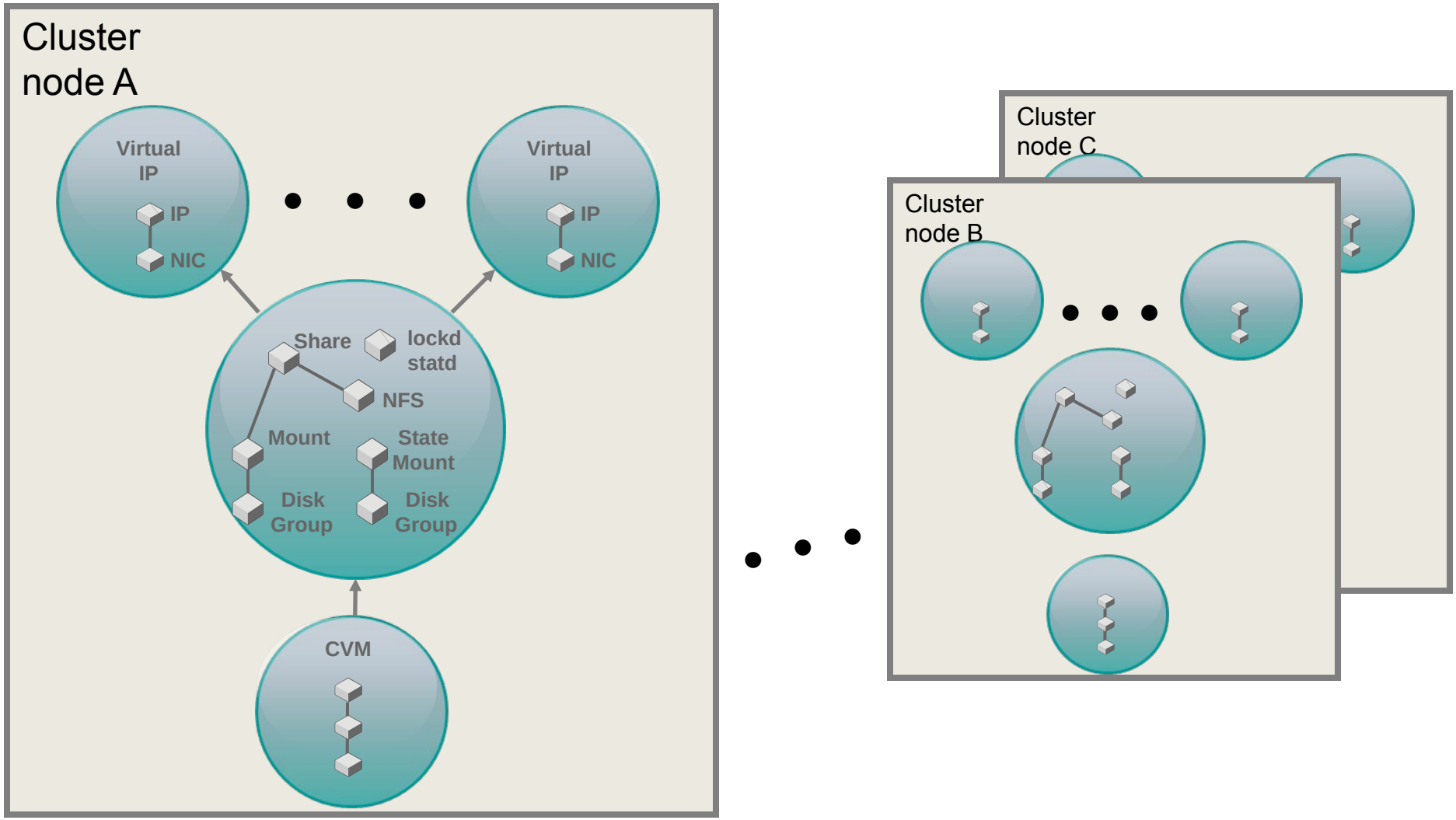
- Forward lock requests from secondary servers
  - Simple to implement
  - Not readily portable
  - May have scalability issues
  - Server to client notifications may be lost or ignored

Our Solution

Fundamentals

- Force status monitor to save state into shared storage
  - A shared file system is created to save state information
  - Each cluster node has a directory in this shared mount
  - The path used by the status monitor is redirected here
  - This data can be accessed by other nodes during recovery
- Configure a service group with these resources in VCS
  - NFS server, lock and status servers
  - File system for saved state information
  - File systems being shared over NFS and matching share resources
- Configure a service group in VCS for each Virtual IP address

Service group layout



Active service management

- Virtual IP address resources are basic management hook
  - All steps tied to this type of resource
  - Each active server has one or more active VIP groups

- Failover processing done by actions attached to IP resource
  - Several points in processing have hooks to allow extra actions
    1. Post-offline (on failed/stopped node after IP is disabled)
    2. Pre-online (on new node before IP is enabled)
    3. Post-online (on new node immediately after IP is enabled)
  - VCS chooses recovery node for this resource, scripts do the work

Initial investigation

- Restart all lock servers during IP resource migration
  - State data manipulated on disk while lock servers down
  - Newly started servers notify clients based on manipulated data
- This allowed lock requests to be granted improperly
  - Race condition exists – locks are released during server shutdown
  - One server may be granting locks after another has released them

Pause lock requests

- No locks granted during lock server shutdown
  - This avoids race condition
  - Lock requests get retried by clients
- Implemented inside file system
  - File system is final arbiter of lock requests
  - Cluster nodes communicate and propagate this state
  - Synchronizes blocking locks with lock request handling
- New lock servers started after lock request handling resumed
  - Reclaim lock requests must be handled normally by file system

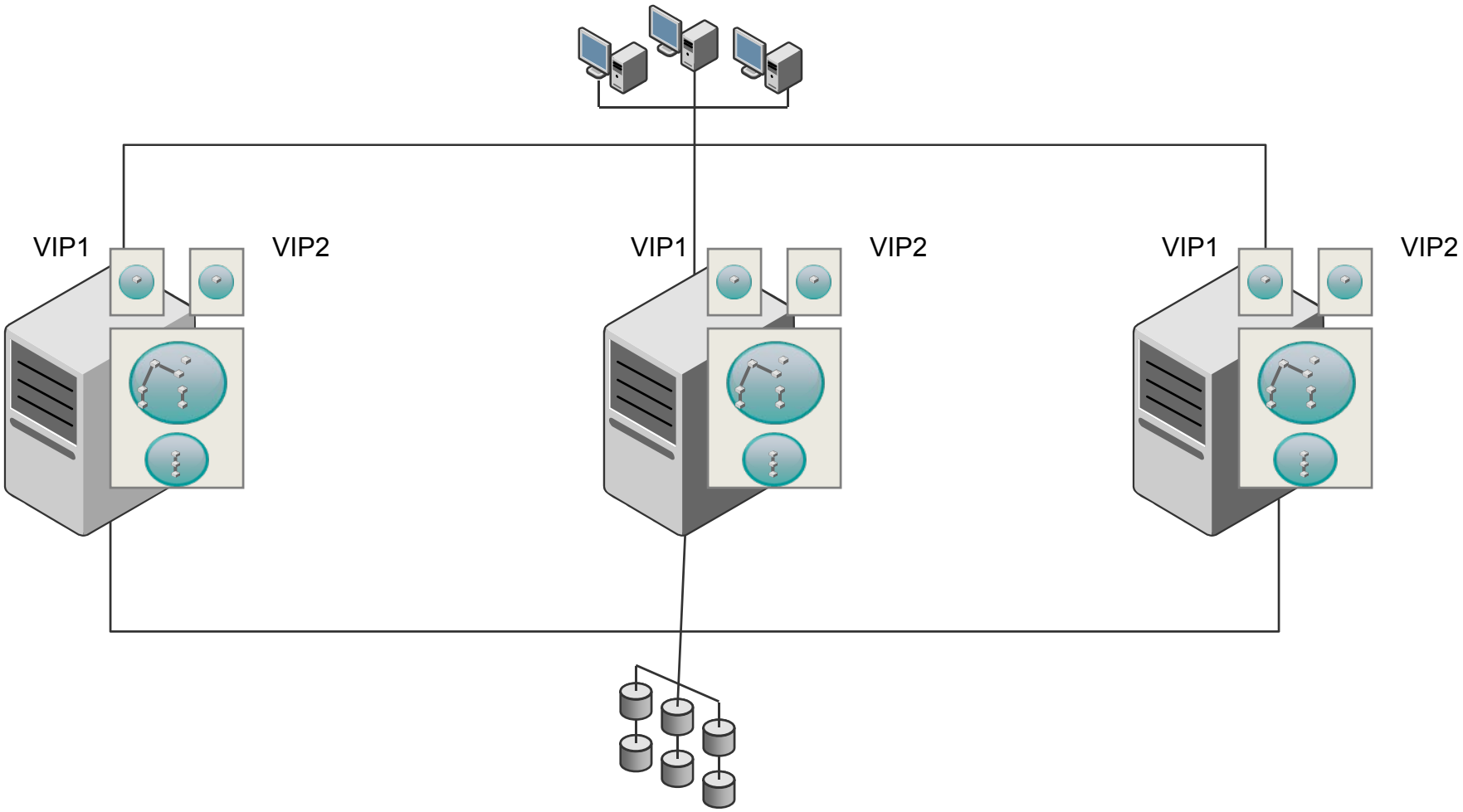
Track nodes requiring lock cleanup

- Cluster nodes track which nodes are using lock management
  - Cluster-wide map of active nodes
  - Feature startup registers with file system
- Node failure can force immediate processing
  - A failed node is checked against the map of nodes sharing over NFS
  - If failed node was actively sharing, start blocking locks immediately

Failover procedure

1. Detect failure (or administrative request)
2. Cleanup for failed/stopped server
    1. Un-register leaving node with kernel lock management
3. Pre-online steps
    1. Register node with kernel lock management
    2. Pause lock requests
    3. Stop lock and status servers on all running systems
    4. Manipulate state tracking data in shared storage
    5. Resume handling locks
4. Activate virtual IP address on new system
5. Post-online steps
    1. Start lock and status servers on all running systems

Common Setup



Future Work

Coming soon to customers

- Current product capabilities:
  - Veritas Storage Foundation Cluster File System 5.0MP3
    - Supports one active NFS server per cluster with full failover
    - Supports NFS serving from all cluster nodes without lock failover
  - Veritas Storage Foundation Scalable File Server
    - Soft-appliance form factor built using Storage Foundation
    - Uses single lock server and lock request forwarding

- Upcoming products:
  - Veritas Storage Foundation Cluster File System (upcoming release)
    - Implements this design to enable NFS serving from all cluster nodes
    - Implementation has been validated during product testing

NFS v4 issues

- New protocol design
  - Full implementation in one protocol
  - More messages involve server-side state tracking
- Different state information
  - Uses timed-lease model for state tracking
  - Clients must regularly contact server to maintain state
- New implementation
  - Generally implemented as a single process
  - Separate persistent state storage from older NFS versions

Q & A