GFS and Friends

Ken Preslan
kpreslan@redhat.com

http://sources.redhat.com/cluster/
http://www.redhat.com/software/rha

Slides:
http://people.redhat.com/kpreslan/gfs-fosdem05.pdf
Many Parts

- GNBD -- Block over IP driver
- CCS -- Cluster Configuration
- Fence -- I/O Fencing
- GULM -- Centralized Lock Server
- CMAN -- Distributed Cluster Manager
- GDLM -- Distributed Lock Manager
- CLVM – Cluster Volume Manager
- GFS -- Cluster Filesystem
- Userspace application failover server
Traditional File Server

- NFS/CIFS/Apache
- Server a bottleneck
- Server a single point of failure
- Upgrade path
  - Replace with a bigger server ($$$)
  - Add another server (Data replication problems)
Block-based interconnects

- Fibre Channel
- FireWire
- iSCSI
- GNBD
- HyperSCSI
- Many others
Symmetric Shared-Disk Architecture
Possible Applications

- Web (http, ftp) serving clusters
- NFS/CIFS serving clusters
- Shared root clusters
- I/O-intensive scientific compute clusters
- Software build clusters
- Parallel Databases (Oracle)
GFS Basics

- Symmetric
- Journaled (1 journal per machine)
- 64-bit
- Works like a local filesystem but inter-machine locks are acquired as operations occur
- Goals
  - Flexibility in terms of locking
  - Flexibility in terms of block transport
GFS vs a Local FS

- Locality not always a good thing
- Deadlock ordering for locks affects a lot
- Journaling/replay more complicated
- Filesystem tree walking has more overhead
- Avoidance of central data structures
  - Inode tables
  - StatFS info
- Some Relaxing of semantics: Fuzzy-ness of quotas, atime, StatFS(soon)
Internal Layout

Glock Layer

GULM

To Lock Servers

DLM

CMAN

Messages to other nodes

Volume Manager

GFS

VFS

User Space

To Lock Servers

Messages to other nodes
Older Features

- Asynchronous journaling
- Multiple journals – one per node
- ExHash directories
- Online Growable (data space and journals)
- Lock caching
- Full read and write-back caching
- Dynamic Inodes
- 64-bits everywhere
- Deadlock avoidance through lock sorting
New Features since 4.2

- Asynchronous locking
- Quotas
- Extended Attributes
- ACLs
- Shared locks are shared between processes
- Multi-writer Direct I/O
- Improved unlink/deallocation
- Improved allocation algorithms
- Improved flock/fcntl()-lock code
New Features since 4.2

• Journaled data
• FS quiesce support
• Better response to memory pressure
• Better transaction/log code
• Ability to convert metadata blocks back to data blocks
• Better NFS support
• Coherent shared mmap() support
New Features since 4.2

- Context Dependent Path Names
- Lots of bug fixes
- Lots of cleanup
Asynchronous Locking

• Lock modules and glock layer rewritten to support async locking
• Glock layer calls into the LM with request
• LM issues a callback with result
• Allows speedups due to parallelization of lock requests
Async Locking (Glock)

- Two options:
  - Prefetch
  - The calling code passes in a structure that defines the request. That structure can be polled or slept on

- Main users:
  - Prefetch inode locks on readdir
  - Statfs
  - Optimization acquiring multiple locks
  - Unlock
Quotas

- Quotas are fuzzy
- Overruns are tunable
- Trade-off: More accuracy means more contention
- User and Group quotas
- Usage limit and Warn limit
Quotas

- Current quota values are cached in lock LVBs (to minimize quota file reads)
- Quota changes are cached in the filesystem in per-node areas
- Changes are synced back to the quota file periodically
- Changes are also synced more often when the user gets closer to their limit
- Idea is to decouple quota handling as much as possible from the quota file
Withdraw

• A new way for a machine to leave the cluster
  - A machine stops all new I/O, waits for pending I/O to complete
  - Calls into lock module with withdraw command
  - Another node does all recovery steps but fencing
• Allows a machine seeing critical I/O or consistency errors to stop accessing the filesystem
• Replaces way over-used panic calls
Lock Module Interface

- Lock_harness – lightweight, GFS-specific CI switch
- Very lock-centric (VCDLM subset)
- Minimal cluster management
- Mount, unmount, lock/LVB operations, plock operations, withdraw, a callback (completion, blocking, recover-journal), recover-journal-done
- Maps Journal ID to nodes
- Handles all GFS inter-node communication
GULM

• Centralized lock/cluster manager
• Up to 5 redundant servers
• Handles membership, quorum, fencing, and locking
• Very GFS-specific
• Older, stable code
• May offer better performance in very large compute clusters
CMAN + GDLM

- Modularization allows us to expose the clustering/locking support developed for GFS for other systems to use as well (CLVM, other CFSs)
- Don't expect that CMAN+GDLM will be useful for everyone, but willing to work to make it useful for others
- Can be used independently of CLVM or GFS
- Newer than GULM, still testing
CMAN

• Cluster Manager
• Heavy VaxCluster influence
• Membership events
• Quorum
• Start/Stop of core cluster services
• Accessable from kernel and user space
• Currently in-kernel, moving to user-space soon
CMAN

- Part 1: Cluster Manager
- A cluster has a unique name and ID on the network
- Multiple clusters can exist on network
- A node can only join one cluster
- All nodes broadcast/multicast heartbeats
- First node to detect a failed heartbeat begins a transition to remove failed node
- Multi-step transition makes sure all nodes are in agreement over new membership
CMAN (quorum)

- Each node in cluster gets a number of votes
- Cluster has an expected number of votes
- CMAN determines if cluster is quorate
- Other subsystems can use this to regulate operation
CMAN

• Other systems can get a list of current members (name, nodeID, IPaddr)
• Other systems can register for callbacks for membership changes
• Available to kernel or user space
Service Manager

- Basic cnxman API not sufficient for GFS or DLM
- Requirement for layered recovery
- Requirement for GFS/GDLM to suspended on all nodes before any node does recovery
- Requirement for GDLM to complete recovery on all nodes before GFS restarts on any node
- Second part of CMAN required: the Service Manager
- For core services only – Userspace servers (apache, NFS, etc..) handled elsewhere
Service Manager

- Symmetric
- Managers which nodes in the cluster are using a particular GFS or GDLM LS
- Represents each GFS or GDLM LS generically as a “Service Group”
- Manages nodes joining or leaving SGs
Service Manager

- SGs are layered for recovery order
- Link from cnxman to SM is: cnxman tells SM when a node fails, SM starts recovery for any SMs the failed node was in
- Members of a SG are all stopped before recovery is started
- All SGs at a certain level complete recovery before any SGs at a higher level are started
- SM factors a lot of cluster management detail out of individual symmetric services and handles it generically for them.
GDLM

- Looks similar to DLM in VMS clusters
- Supports many independent lock-spaces
- Nodes “join” a lock space to begin acquiring locks
- Runs entirely in the kernel
- Heavy use by GFS makes userspace DLM impractical (performance/latency, memory, callbacks)
- Depends on CMAN for cluster management
Fencing

- Generic infrastructure to support I/O fencing
- Pluggable agents to support different hardware
- About 20 agents currently
- Various different methodologies
  - Power cycling
  - Fencing in I/O path (fabrics, switches)
  - Fencing of I/O device (iSCSI, GNBD)
- “Easy” to add another method
Fencing

- **GULM:** GULM master server fences dead client
- **CMAN/GDLM:**
  - Fence system for CLVM/GFS is a simple userspace daemon controlled by CMAN (SM)
  - Fencing is also symmetric (any node can fence any other)
  - Using input from CMAN/SM, fenced decides who needs to be fenced
  - Fenced is just a SM Service Group
  - Fenced registers at the lowest level so a node is fenced before DLM/GFS recovery happens
CCS

- Cluster Configuration system
- XML-based configuration files
- Mostly there to define fencing methods for nodes
- Configuration files replicated and kept in sync on all nodes of cluster
- No longer requires shared storage
CLVM

- A userspace daemon layered on top of LVM2/DM
- Uses CMAN+GDLM (GULM?)
- Working on clustered mirror and snapshot targets
Future Work

• Big short-term targets
  - Small file performance
  - FSCK speed improvements
  - Local storage utilization
• Shared Root
• File locality controls
• Metadata locality to specific devices
• Block based file backup support
Future Work

- B-Trees instead of allocation bitmaps
- File System shrink
- Dynamic hotspot elimination
- Forced Unmount
- I/O load balancing
- Range-level locking
- DMAPI?
Future Work

• Buffer Passing
• Operation Passing
• Filesystem Snapshotting
• File Versioning