Replication and Migration

Background, Requirements and Strawman

Migration and Replication

- The point is: Increased availability
 - Through replication of shared read-only data
 - Less "NFS Server not responding"
- The point is: Increased transparency
 - Through transparent migration
 - Eliminate client reboots to bind to new location

Replication and Migration

- Replication is the creation of one or more copies of a "file system"
 - Distinguish "read-only" from single writable master and "read-write" replication
- Migration is the movement of a "file system" from one server to another
 - Useful only when transparent
 - Necessary even when not
 - Rubber meets the sky on migrating single writable file system

NFS V4 Migration/Replication

- NFS Version 4 defines client to server interaction ONLY
 - List of servers hint to client where file system may migrate/replicate to
 - The volatile file handles allow cheesy solutions³
 - Hashed file names persist©

Quick Review

- Sun client failover for NFS Version 3 (and 2)
 - Store pathnames in rnodes
 - Failover transparent with re-resoolve using alternates from Automounter maps
 - No migration support?
 - What replica consistency? ©
- Solves the problem of hung client when "replicate" read-only binaries etc are available

Quick Review

- AFS and DFS replication
 - Single master write copy for replication, read-only replicas
 - Client failover through "file system" resolution using replicated data base
 - "Inodes" preserved (file system semantic)
- Migration leverages infrastructure
 - Move a home directory (writable file system)

Quick Review

- rdist
- rsync

- Transparent client failover
 - For read-only replicas and migrated writable volumes
- Performance
 - Bandwidth conservative
 - Differences propagated
 - Restartable
 - Client lockout time minimized
- Security
 - As good as V4

- Scalable
 - Huge file systems
 - Small file systems
- Capability negotiation between "peers"(?)
 - I believe this referred to things like attribute differences
- Efficient multi-way replication (propagation)

- Correctness
 - "Atomic" propagation of file system from client view
 - Failover to a correct replica version
- TCP/IP based
 - No legacy UDP requirement

- Replica versioning non-existent
 - Failing over to "correct" version of replica impossible?
 - Base V4 protocol change?
 - Proposal: Investigate versioning requirement

- Single vs. multi-master
 - Multi-master entails conflict resolution
 - Proposal: Single master copy sufficient objections?
- Disconnected operation irrelevant
 - Corollary to above
 - Proposal: Disconnected operation for clients not required – objections?

- File oriented
 - Fits with NFS model, and heterogeneous
 - Efficiency concerns
 - Block-oriented not heterogeneous
 - Proposal: Draft proposal for comments.
- Replicating "opaque" (to NFS) local file system attributes
 - Proposal: NFS Version 4 named attributes propagated – unsupported attributes not

- Migration/replication only for V4
 Not a general (rdist) mechanism
- Lock and delegation propagation
 - Certainly a requirement, but ouch!
 - Proposal: Locking and delegation state propagated, acceptable that it resembles a server reboot.
 - (Brent?)

- We pushed need for reliable name/location service from clients to servers
 - Proposal: Investigate how far to tie back end protocol to name sevice.

RFC: NFS V4 "file system"

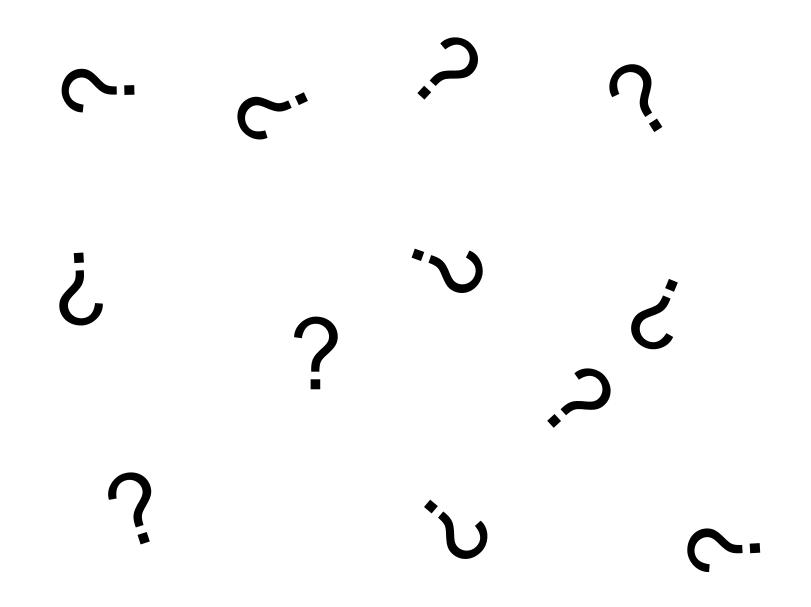
- Has a file system ID
- A closed set of unique "file ids" aka inodes³
- A set of attributes associated with the "file system"
- The basis for replication and migration?

File system model issues?

- fsid should define a "file system"
 - Hard links exist within the file system and must be maintained
- Attributes of file must be maintained
 - Times cannot be screwed up
- Heterogeneous interoperability
 - What happens if you migrate a file system from multibyte to single byte name encoding environment?

The process

- Recommendation is to re-charter the existing workgroup to specify back-end protocol
 - Need to write new charter



Migration in DFS: an example

- A read-only clone is made of a "fileset" (replication unit)
 - Brief operation, copy-on-write properties for primary
- Clone is transferred
 - During transfer clients have read/write access to primary fileset
- The clients are locked against further updates during incremental transfer of new data
- The clients atomically fail over to the new location

Migration in DFS: an example

- Migrated volume only visible on successful transfer
- Client disruption is minimized
- Performance in the face of large files (by doing block incremental completion phase) is solved

Replication: DFS example

- Replication based on clone operation as in migration (and backup)
- Replicas are versioned
- Transaction to enable replica on successful transfer
- Coda extends to writable replicas?

Replication: rsync example

- Super-rdist protocol with recovery
- Over TCP
- Propagates block level updates
- Works on standard file systems
- Could be basis for NFS Version 4 replication
- Seems to lack "atomic" update from client perspective and versioning (to deal with failure recovery)