SANE 2006 – NFSv4 15/05/06 11:03

Motivation for replication

- facilitate wide area interoperation and access
- facilitate file system administration

Solutions

- Global name space
- Migration -> simplify data relocation and load balancing
- Replication -> improve performance and availability (read-only replication is partially supported in v4 with various hooks)

Global name space

NFSv3

- no support for global name space, migration of replication.

Global name space

- **/nfs** is the global root for all NFS file systems
- entries under **/nfs** are mounted on demand
- the format of reference names under **/nfs** follows DNS conventions, eg **/nfs/umich.edu/lib/file1**.
- automatic redirection will push the client in the right direction.

Extended use of DNS

- DNS SRV resource records carry NFS server location information.
- The corresponding name server maps a logical name to some NFS servers
- Client-side utility enables transparent access to the global name space (DNS mod to automounter)
- Support file system replication and migration

When booting **/nfs** is completely empty. Entries get added when servers are accessed.

Mutable replication (r/w)

Consistent updating is hard!

Design goals:

- allow users to modify data at any time.
- read access have no additional cost.
- support strong consistency guarantees
  - sequential consistency
  - synchronized access (read returns the most recent write)

Writing is done on the basis of quorum. All replicas are in read-mode, until a write request comes in. The client contacts one of the replicas, the replica requests write access and only gets it when a majority of the other replicas agrees. When done, the primary sends a close signal to all replicas.

During file modification, clients may read from any server.

Every NFS server keeps track of its siblings. The primary removes from its active view any server that fails to responds to its requests. On file close, the primary also sends its active view to the other replicas. Hence:

- failed servers cannot become primary
- failed servers rejoin the active view after synchronisation

Update distribution

- primary ack's a client write after a majority of replicas reply
- naturally this is a bottle neck
- the NFS i/o daemon buffers data to allow for asynchronous writes
- with parallel updates the response time depends on the median RTT between the primary and other replica servers.

recovery from primary server failure

- detected by either the client or one of the replicas
- the client will do a standard NFS recovery by selecting a new server for its write action
- a replacement primary is elected if a majority of replication servers are active
- the replacement primary retrieves and distributes the freshest copy of the file and reconstructs its active view.
- client will need to re-issue the last failed request only (which it does anyway).

Performance

Adding replicas within the same RTT only has a slight performance hit. Going from 1 to 2 bumps things up a little, but from there on it stays pretty level.

However, the RTT between replicas does have a huge hit. It’s a very good idea to use a quorum-like setup. One on site A, one on site B and one in the middle. Hence the quorum is reached quicker on both sides of the infra.

Hierarchy locking also improves performance hugely. In this case you’re limiting the cost of replication.