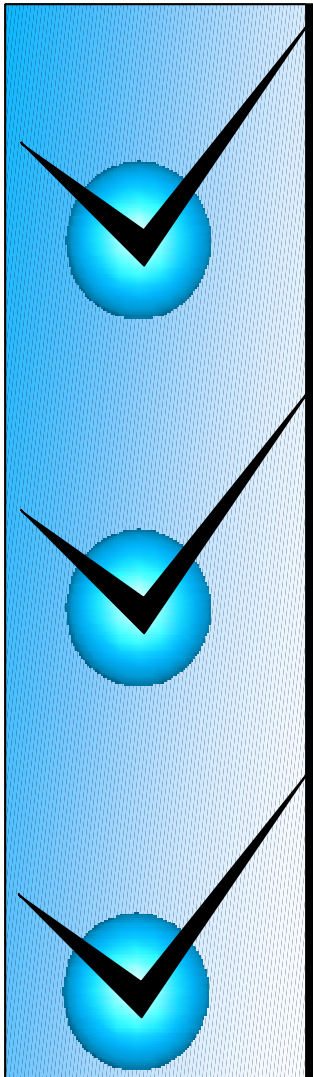


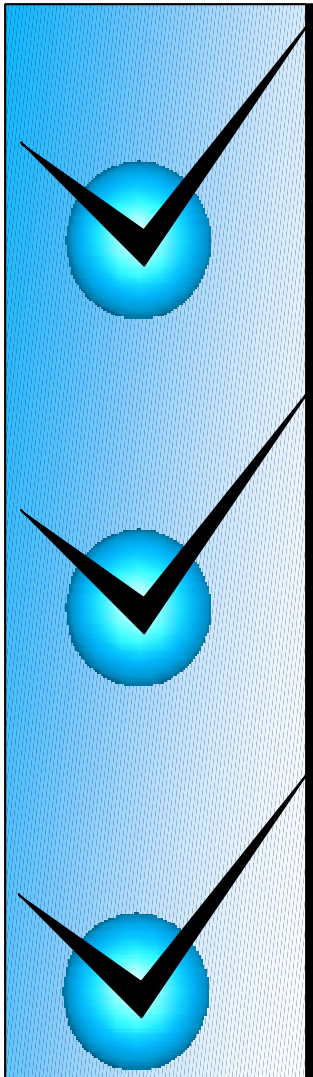
Delegations in NFS-v4

David Noveck

Member of the Technical Staff

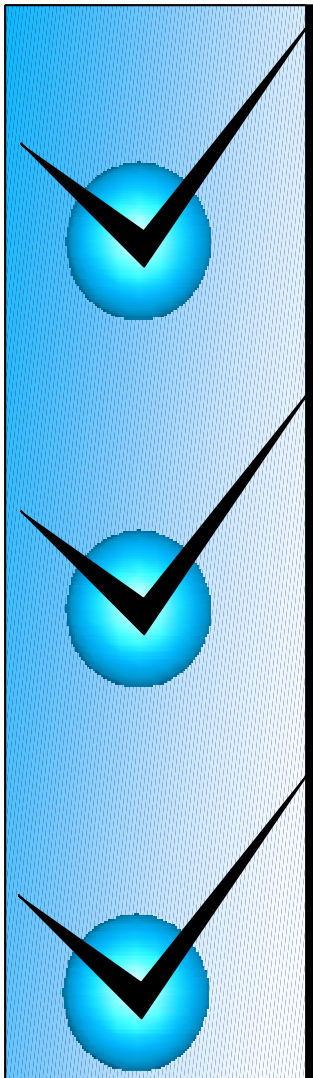
Network Appliance

dnoveck@netapp.com



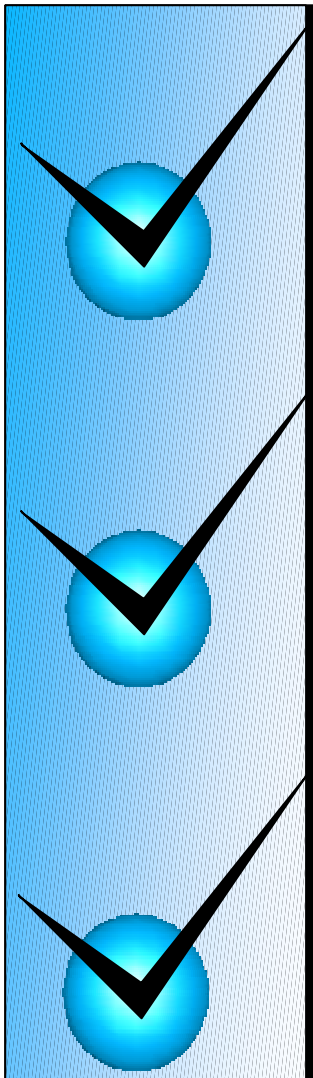
Summary

- Introduction to delegations
- How they work
- Benefits
- Implementation/deployment issues
- Possible future extensions



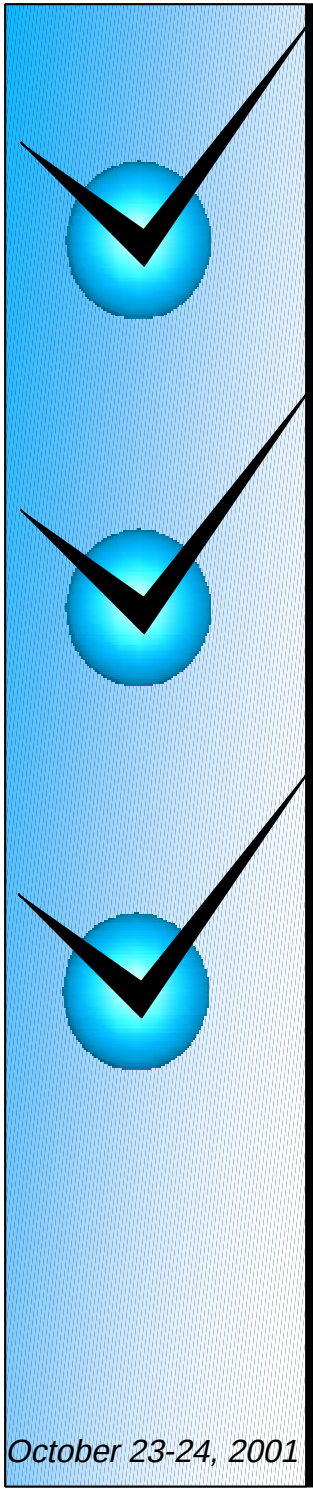
What are delegations?

- A mechanism to reduce latency
  - By performing operations locally
  - When sharing patterns allow that
- While maintaining correctness
  - So server may *recall* delegations
  - Callback RPC's are used



Why delegations in V4

- Allow V4 use when latency is high
  - Also a win in lower-latency environments
  - Depends on ratio of local/remote speeds
- V4 has OPEN and CLOSE
  - Good framework for delegations
  - But try to avoid sending them too often
  - Also locking requests



Types of delegations

- Write
  - Exclusive access by a single client
  - Client arbitrates opens among processes
  - Client arbitrates locks among processes
- Read
  - Shared read-only access by many clients
  - Clients may do opens (for read) locally

Getting a write delegation

- Happens at OPEN time
- Server *may* grant a delegation
  - If no other client has file open
- Client may keep delegation
  - Until server recalls it (via a callback)
  - Client may return it voluntarily

Using a write delegation

- Client knows he is the only user
- Doesn't have to involve server
  - To do OPEN (share reservation local)
  - To do CLOSE (share reservation local)
  - To check modified time
  - To do LOCK, LOCKT, LOCKU
- Client arbitrates among processes

Using a write delegation

- Client may avoid write flush on CLOSE
  - If server exports space reservation info
- Allows flush to be done lazily
- Flush may not be done at all
  - If the file is truncated before flush

Recall of write delegation

- Server recalls when
  - OPEN request (usually from another client)
  - RENAME, REMOVE, SETATTR
  - IO request from another client
- Delayed until delegation return
- No recall on GETATTR
  - Server directs GETATTR callback to client

Getting a read delegation

- Happens at OPEN time
- Server *may* grant a delegation
  - If no client has file open for write
  - And no client has file open denying read
- Client may keep delegation until recall or voluntary return

Using a read delegation

- Client knows nobody is writing
- Doesn't have to involve server
  - To do read-only OPENs
  - To do corresponding CLOSEs
  - To check modified time

Recall of read delegation

- Server recalls when
  - OPEN request for write
  - OPEN request denying read
  - RENAME, REMOVE, SETATTR
  - WRITE requests
- Delayed until delegation return

Delegation recall process

- Server does RECALL RPC
  - Client replies
- Transfer state to server
  - Do deferred OPENs
  - Do deferred CLOSEs
  - State transfer for write delegation
- Return delegation

Delegation recall process

- Locking state transfer
  - Special LOCK request
- Transfer modified file data
- Effect deferred truncation

Estimating the benefits

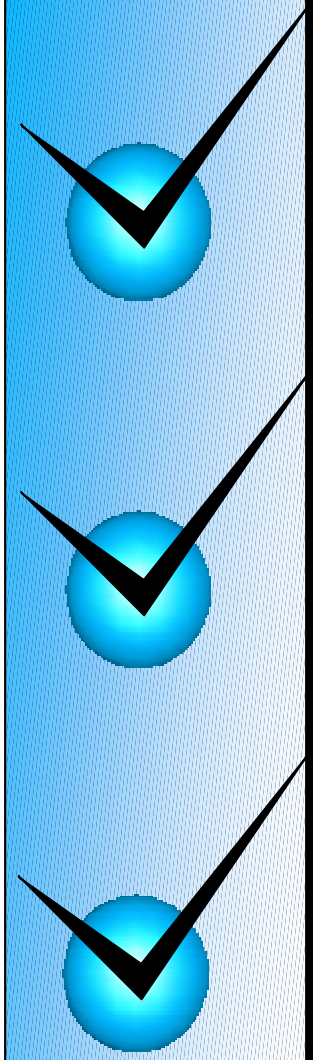
- Greatest when,
  - Frequent OPENs and CLOSEs
    - Generally small file environments
  - When file locking is used
- When sharing is either,
  - Not intense
  - Read-only

Estimating the benefits

- Greatest when,
  - Latency is high
  - Client is *very fast*
    - e.g. Application-integrated user-mode client
  - Server is heavily loaded
    - Many clients
    - Lots of intense read sharing

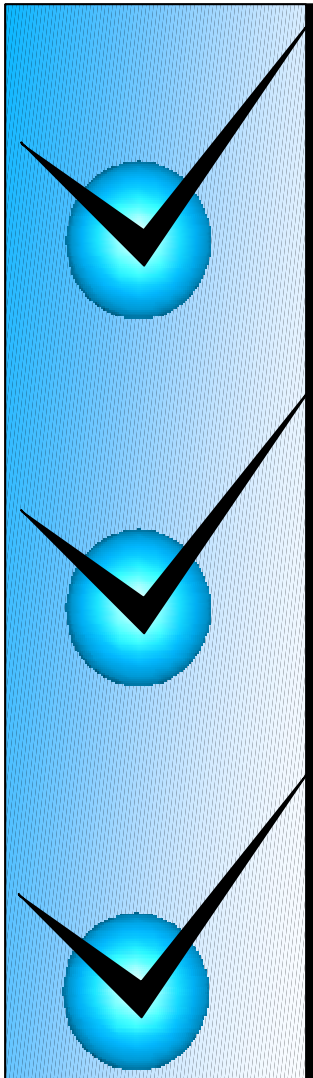
Where are they now?

- In specs
  - RFC 3010
  - Also DAFS 1.0
- In RFC 3010 successor
  - Possible changes to deal with NAT and firewalls
  - Some clarifications



Implementations?

- Limited implementation work so far
- Very limited testing at last bakeoff
  - Initial delegation handoff (without a panic)
- Should get farther at this one
- Implementation has lagged
  - Getting the old features working
  - Delegations is a new direction



Deployment issues

- Delegations are optional
  - Server can just not implement
  - Client can return immediately
- Makes it easy to not implement
- No benefit unless both have it
  - Need to get a critical mass
  - Will make delegations a big win for V4

Long-lived delegations

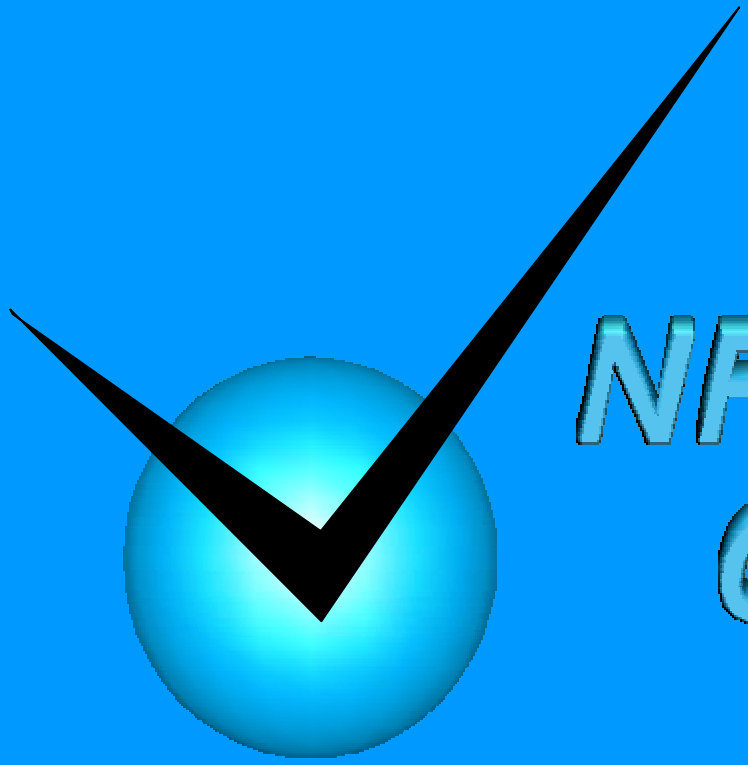
- Keeping delegated files on disk
  - When distant from server
  - Particularly in the proxying case
- Needs delegation re-establishment
- In spec, but if nobody implements
  - Could be lost at Draft Standard
  - Might come back in a minor version

Directory delegation

- Delegation of directory contents
- For READ, is a straightforward protocol extension
  - Avoids frequent revalidation
  - Reduces server load
- WRITE delegation is harder
- Possible extension(s) for V4.1

Directory-tree delegation

- Further extension
- Potential for good performance
  - Even when latency is *very* high
- Hard links are a big issue
  - Directory tree becomes directory DAG
- Merits investigation for a minor version



NFS Vendors Conference