Stephan Wiesand
November 19, 2002

Remote Filesystems at DESY

- how do NFS, CIFS, AFS compare ?
  - and what the heck is CIFS anyway ?
  - what's the difference between NFS V2/V3 ?
- what else is out there ?
  - DFS, Coda, Intermezzo, DFS, ...
- what will the (near) future bring ?
  - NFS V4, ongoing work on CIFS & AFS

A Good Remote Filesystem

- is easy to use
  - looks & behaves like local FS
- is easy to deploy and maintain
- is secure
- is (at least nearly) as performant as a local FS
- scales well
  - to large numbers of servers & clients
  - to large amounts of data
- deals well with outages of underlying services
- and hence does not exist.

Agenda

- Common concepts
- The evolution of NFS
  - V2, V3, V4
    - V3 with system based authentication is reality
    - anything else is fiction !
- CIFS
  - much V4 functionality
- AFS
  - much V4 functionality
- Other filesystems

Common Concepts

- for Performance
  - avoid overhead
    - few requests to server per operation
  - client side caching
    - problem: cache coherence
    - needs mechanisms like locks, delegation, callbacks
- for Scalability
  - replication
    - of data or servers
  - relocation
    - of data between servers

The Evolution of NFS

- 1989
  - RFC 1094
    - NFS V2
    - describes NFS & Mount protocols
- 1995
  - RFC 1813
    - NFS V3
    - nominally minor enhancements
- November 2002
  - Draft RFC 3010-bis-5
    - expires March 2003
    - protocol not finalized
    - NFS V4
    - major enhancements
    - very different from V2/V3
    - reference implementation underway

NFS V2

- design goal: stateless protocol
  - simple servers,
  - works well on unreliable networks & computers
- client simply retries until it receives a reply
  - connectionless UDP ok
- requires operations to be stateless and idempotent
- alas, many file(system) operations aren't
  - -> additional protocols, not part of core NFS
  - -> operations history kept on server/client ?

NFS V2

- examples:
  - "mount filesystem"
    - -> mount protocol
  - "lock file"
    - -> Network Lock Manager (NLM)
  - "delete file"
    - server may remember recent operations & ignore duplicates
- problems:
  - now already 3 protocols, dynamically assigned ports
    - hard to tunnel firewalls
  - what if repeated requests are intentional but arrive out of order ?

All NFS V2 Operations

- NULL (noop)
- get / set file attributes
- get filesystem attributes
- read symbolic link
- read from / write to part of file
- create / remove / rename file
- lookup file / directory
- create hard link
- create symbolic link
- create / remove directory
- read directory

NFS V2 Performance

- high overhead
  - primitive operations
  - paths dissected by the client
  - reading /foo/bar/baz:
    - "lookup foo"
    - "lookup bar"
    - "lookup baz"
    - "read from baz"
- no client side caching defined by protocol
  - clients typically cache for O(1s), no validation
- all operations defined to be synchronous
  - some servers did asynchronous writes all the same

NFS V2 Security

- underlying RPC protocol allows several different schemes for authentication
  - including Kerberos (!)
  - but not for the mount protocol
- in practice: only system based authentication
- no authentication of server to client
- permissions checked by numeric UID/GID
  - assumed to be equal on server and client
- protocol allows
  - "squashing" ID 0
  - readonly exports

NFS V2 Summary

- simple servers, lean protocol
- but smart clients
- is easy to use, deploy, maintain
- works in shaky environments
- not completely reliable, by design
- slow
  - primitive, synchronous operations only
  - no real caching
- does not scale well
  - no replication
  - server has to do everything, in small steps

6 Years later: NFS V3

- supports 64 bit file sizes and offsets
  - allows files > 2 GB
- no more artificial limits on lengths of path and file names
- new operation for access check on the server
- character & block special files, mknod operation
  - for device files
  - allows diskless clients
- no more limit of read & write request size
  - limit is specified by server

NFS V3 Enhancements continued

- server returns more information on every call
  - saves extra inquiries
  - helps validate cache
- more complex operations, like "readdirplus"
  - saves many lookups
- asynchronous write & modify operations and a "commit" procedure
  - V2: do, wait, do, wait ...
  - V3: do, do, do, commit
- some new operations providing hints to the client for cache validation

Summary: NFS V3

- is still stateless
- is no more secure than V2
- has fewer limitations
- still has no real client side caching
  - works better, but
  - still weak semantics
- has significantly reduced overhead
- in reality, makes a big difference
  - amazingly fast
  - more active clients per server possible
  - newer implementations may just be better ...

NFS V4: ETA Q3/2003

- single, stateful protocol
  - including lock & mount
  - OPEN & CLOSE operations
  - TCP transport only
  - no retries
  - smarter servers
  - simpler clients
  - -> WAN, firewalls ok
- enhanced security
  - principal based (strings, not numeric Ids)
  - negotiated
  - uses RPCSEC_GSS
  - Kerberos 5 mandatory
  - other protocols available for GSS-API optional

NFS V4 Enhancements continued

- UTF-8 file names
- protocol support for Access Control Lists (ACLs)
  - well defined
  - Windows, not POSIX semantics
  - optional
- protocol support for data replication & relocation
  - not so well defined
  - leaves much to the implementation
  - will probably not work between different vendors' servers
  - optional

NFS V4 Performance enhancements

- reduced overhead
  - compound statements
  - multicomponent lookup
- client side caching
  - persistent
  - w/ strong semantics
    - locks
    - share reservation
    - delegation
- locks are leased
  - on request by client
  - byte range locks, like V2/V3
  - mandatory, unlike V2/V3 (advisory)
  - for defined lease time
  - can be renewed
  - expiry considered failure

NFS V4 Share Reservation & Delegation

- Share Reservation
  - file level
  - is not a lock
  - requested by client on OPEN
  - denies OPEN to other clients (read and/or write)
  - lasts while file opened
  - revocable (callbacks)
- Delegation
  - may be granted by server on OPEN
  - can not be requested
  - client has all responsibility for file
    - locks may be served locally, ...
  - revocable (callbacks)

Summary: NFS V4

- will be
  - fast
  - reliable
  - scalable
  - versatile
  - secure
- ... once it's fully implemented ...
- it will no longer be simple
- it's replication is
  - read-only
  - not likely to become available anytime soon
  - not likely to be interoperable

CIFS

- the protocol formerly known as SMB
  - Server Message Block
  - by IBM in mid-80s
  - more than a filesystem
    - sharing printers
    - & other resources
- when SUN invented "WebNFS" ...
- ... Microsoft renamed SMB to "Common Internet FileSystem"
  - and started adding undocumented enhancements to it
- 01/2002: CIFS Specification V 1.0 by SNIA (www.snia.org)

CIFS

- SNIA: "Storage Networking Industry Association"
- Microsoft is "Associated Member" (membership 4th class)
  - recently started using term "Microsoft SMB" for a superset of SNIA CIFS ...
- "Truth is what Windows boxes do"
  - G. Carter, Samba/HP
- CIFS is Windows' native remote FS
- Samba is a free Unix server/client
  - Linux has smbfs client
    - 2.6 has a new cifsfs

CIFS Features

- file and record locking
  - mandatory
- client side caching with strong semantics
  - not persistent
  - opportunistic locks
    - similar to NFS V4 delegation
    - synchronous operation as fallback
- file change notification
- batched operations
  - like NFS V4 compound statements
- data replication
  - implemented ?
  - not in Samba

CIFS Features continued

- Unicode filenames
- extended attributes
  - Windows / MacOS secondary data streams
- assumes a reliable, connection-oriented transport
  - TCP, NetBios over TCP
- principal based authentication
  - many protocols
  - including Kerberos 5
- ACLs
- DFS
  - an automounter for CIFS

Summary: CIFS

- is a full featured file system that
  - should perform well and be reliable
  - is scalable
  - can be used securely
- is actively pursued by major companies in an open fashion
  - IBM, HP, ...
- has a few drawbacks:
  - no full Unix implementation yet
    - but Samba 3 & cifsfs very promising
  - weak standardization
    - no guarantees that Microsoft will play by the rules
    - and NAS vendors ?

AFS is different

- NFS & CIFS export a local filesystem to their clients
- AFS is purely remote
  - has it's own local format
  - even on the server, access is only possible through the client
- advantage: AFS is not restricted by limitations of the server operating system
  - ACLs are available on all combinations of client & server OSs
- disadvantage: effort to switch

AFS Features

- Client side caching
  - persistent
  - directory level delegation through callbacks
  - flush on close
  - files are "uniqified" by a number incremented on every change
    - cheap validation
- comes with it's own authentication system
  - principal based
  - Kerberos 4 (K5 works)
- data organized by volumes
  - replication (readonly)
  - relocation
    - live, on the fly !

AFS Features continued

- native volume backup
  - max restorable volume size is 2 GB
- support for heterogeneous environments
  - "@sys" in paths is substituted by the "sysname"
    - list of those RSN
- OpenAFS: a free, open source implementation
  - available for all major platforms
  - here and now
  - with all the features
  - growing user community

AFS Drawbacks

- file level locking only
  - no byte range locks !
    - not anytime soon
  - bites many Windows applications
    - MS Access
    - MS FoxPro
    - ...
- directory level permissions only
  - unix mode bits have unusual semantics
- complex to set up and maintain

Summary: AFS

- has almost all desirable features
- only lacking
  - readwrite replication
  - byte range locks
    - not anytime soon
  - disconnected operation
    - 2003 ?
    - not even on NFS/CIFS's agenda
- fully available now
- proven to work
- free
  - but license prevents inclusion into kernels
- but
  - complex
  - not perfect for Windows

Other Remote Filesystems

- Coda
  - descendent of AFS 2 (we use AFS 3)
  - readwrite replication
  - disconnected operation
  - secure authentication
  - slightly experimental
  - slow writes
  - bugs at large scale (?)
- Intermezzo
  - "a better Coda"
    - faster
  - disconnected operation
  - experimental
  - no security yet
  - development seems stalled

More Remote Filesystems

- DFS (the real one)
  - part of DCE (Distributed Computing Environment)
  - AFS on steroids
  - seemed to be the future a few years ago
    - no longer
  - available for linux ?
  - very complex
- GFS
  - commercial, linux only
- Cluster file systems
  - GPFS, ...
- Strange ones
  - OIF
    - Oracle Internet FS, stores files in DB
  - WebDAV

Summary

- NFS V3 will be with us for a while
  - insecure, not perfect, but working, fast, easy
  - linux V3/TCP RSN
    - may make it more reliable
    - secure tunnels
- Looking forward to NFS V4 – next year ?
- AFS will still be the most important filesystem during 2003, at least on Unix
- CIFS may become a strong contender - if not hogged by MS
- If we simply buy NAS, do we still need to care what's inside ?