SMB3 and Linux
Seamless POSIX file serving

Jeremy Allison
Samba Team

jra@samba.org

Opening Windows to a Wider World

Isn't cloud storage the future ?

Yes, but not usable for many existing apps.

Cloud Storage is a blob store

- Blob stores don't map very well onto the open/read/write/close random access semantics of most applications.
- Apps are changing to cope with no random access semantics of cloud stores, but this will take time.

We still need file access protocols

- Even running in the cloud, pointing existing apps at file servers is useful.
- Only two viable options – NFS (v4) and SMB2+ (known as SMB3 from now on).
- Why SMB3 and not NFSv4 ?
  - It's the clients..
- Both NFS and SMB are supported by the only clients that matter, Windows MacOS X and Linux.
  - But Windows supports SMB3 much better than NFS

SMB3 vs NFSv4

- Roughly comparable.
  - SMB3 has more features.
- NFSv4 includes:
  - Delegations – file and directory (SMB2 leases)
  - Name spaces (MS-DFS)
  - Sessions (long-lived handles)
  - Adapted SMB ACL model (disaster)
  - Parallel NFS (pNFS)
  - Defined over RDMA

SMB3 vs NFSv4

- SMB3 includes:
  - Transparent failover
  - Clustering (Active/Active shares)
  - SMB over RDMA
  - Multichannel (multiple NIC)
  - Encryption
  - Leasing files/directories
  - Snapshots
  - Server-side copies
- Rapid development (whatever Microsoft adds next).
- Windows clients *really* want to use SMB3.

SMB3 vs NFSv4

- NFSv4 has one current advantage in Linux → Linux environments:
  - Close to POSIX semantics.
  - Designed around POSIX clients → POSIX servers.
    - Advisory locking, rename open files, unlink open files etc.
    - Extended attributes and other things added later.
  - Modifications for Windows clients are add-ons.
- How do we fix this for SMB3 ?
  - SMB3 UNIX extensions !
  - SMB3 is really close to what we need for Linux → Linux.
    - Add POSIX semantics to a Windows protocol.

Enter flexible Samba

- We have a history making this work.
  - SMB1 “UNIX extensions”.
- Originally created by old (non-insane) SCO and HP.
  - Method of adding POSIX 'info levels' into SMB1 query/set file info requests.
- Later extended by Samba for both client and server:
  - POSIX pathnames
  - tranport level encryption
  - POSIX ACLs
  - Symlinks
  - POSIX behaviors (rename & delete, file locking)

Opening Windows to a Wider World

SMB1 Unix Extensions

Client

Server

Negprot req:

Negprot reply:
UNIX capabilities

SetFSinfo req:
UNIX bits I want

SetFSinfo reply:
UNIX bits I will support

UNIX specific req:
Open with POSIX
Pathname /foo/bar

UNIX specific reply:
Open file handle

What SMB1 Unix Extensions got wrong

- Horrible hack job - abusing the protocol to add elements it was never designed to do:
  - Tridge: “Using SetFSinfo to set global state on the protocol connection makes me want to vomit !”
- Apple ended up doing the same thing by adding Macintosh share-specific info levels for get/set.
- Biggest problem was setting the server “global state”.
  - Once UNIX extensions were negotiated existing operations are expected to change behavior.

(More of) What SMB1 UNIX Extensions got wrong

- Symlinks and security - this is a disaster zone:
  - Windows clients want to follow symlinks on the server.
  - UNIX clients MUST NOT follow symlinks on the server.
- Transport level security (SMB1 encrypt) poorly designed.
- Extended Attributes (EA's) differences ignored.
  - Windows EA's are not a good match.
- No other server than Samba implemented them.

SMB3 UNIX Extensions – A Clean Slate

- SMB3 is entirely handle-based.
  - The only pathname operation is to use “Create” to turn into a handle.
- Handles collect all the properties needed to implement
- POSIX semantics into one place.
- Handles are used for delete/rename/locking/extended attributes.
  - All the areas where POSIX requirements differ from Windows.

SMB3 Create Contexts

- SMB3 has an in-built mechanism to extend the pathname → conversion: Create Contexts.
  - Create contexts are named “blobs” of data attached to the “Create” request and reply.
  - Unknown create contexts are ignored.
- Create contexts allowed Microsoft to extend SMB2 → SMB3 features by adding named elements to “Create” operations.
  - Examples include “TWrp” (Timewarp) snapshot request and “SMB2_CREATE_APP_INSTANCE_ID” request (identified by a GUID).
- A create context named “POsx” (or more likely a GUID) will do nicely to add POSIX features to a create

How to negotiate SMB3 UNIX Extensions ?

- SMB1 Unix extensions used a “POSIX CAPABILITY” bit in the 32-bit capabilities field in the initial server negotiate response.
  - Required coordination with Microsoft.
  - Could be re-used for SMB3 (bit already allocated).
- SMB3 has an in-built mechanism to extend the negotiation of client → server capabilities.
  - Modeled after SMB3 Create contexts, but done at SMB3 initial negotiate time.
  - Not a GUID (missed opportunity IMHO) – a 16-bit field. *Still* have to coordinate with Microsoft :-(
- Do we need UNIX extensions negotiation at all ?

The Apple Solution

- Apple used a similar method to add
- Mac-specific features:
  - AAPL create context (implemented in Samba by vfs_fruit).
- AAPL isn't a very clean design.
  - Modifies contents of returned info-levels once negotiated.
  - Negotiation step done on an initial Create call on a name of “” in the share.
  - Reproduces the sins of SMB1 Unix extensions (global server state turned on by a single request).

The (proposed) Samba Design

- Minimal (or absent) protocol negotiation.
- No negotiation on features at Create time:
  - This lead to lots of complexity in the SMB1 code.
  - Add new create context for pathname → handle creation.
  - Use existing Windows pathname parsing (UCS2, not UTF8). No alternate data stream names.
  - Server gives “all or nothing” POSIX semantics if context returned.
- New handle flagged as “UNIX” internally, all operations become POSIX on this handle.

The (proposed) Samba Design

- What are the POSIX semantics on a handle ?
  - Reads/Writes ignore POSIX locks (not Windows).
  - Lock requests become advisory (not mandatory).
  - Unlinks/renames are allowed on open handles (if no other non-UNIX handles open on the same file).
  - Directory listings return POSIX namespace.
    - Should QueryDirectory change info level returns ?
  - Get/Set EA's use UNIX not Windows namespace.
    - Do we expose the user. / system. or other EA namespaces ?

The (proposed) Samba Design – Unsolved Issues

- Symlinks are still a problem.
  - How do we create them ?
  - What are the EA and ACL operations permitted on them ?
- POSIX Info levels are 2-bytes (0x200 – 0x2FF).
  - Few used (00 - 0B), but won't fit into existing 1-byte SMB2+ info level space.
  - As set/query info levels are attached to a file handle, we could define extra info levels only on POSIX handles.
  - Use FSCTL calls instead for extra POSIX requests ?
- Windows lock ranges are unsigned, POSIX are signed.

Implementing the SMB3 UNIX Extensions in Samba

Opening Windows to a Wider World

- Have already been prototyped by both Volker Lendecke and Richard Sharpe of the Samba Team.
  - Internal Samba issues prevented this code going into production.
- 'Global' state finally removed from Samba git master branch March 2016.
  - Removed the evil 'lp_posix_pathnames()' global call from the Samba VFS.
  - 'POSIX' flag on a handle now the only required state to determine server operation.
- Still some cleanup to do to expose all the Linux → Linux operations over SMB3, but mostly done

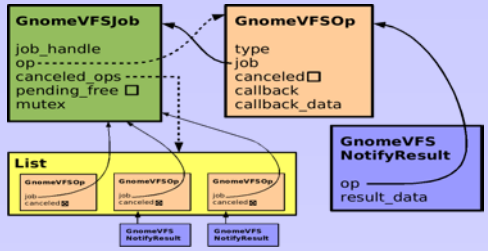
Implementing in Samba: The ACL Problem

- SMB3 natively uses Windows ACLs
  - Similar but not the same as NFSv4 ACLs.
- Linux uses POSIX ACL draft spec, coded up by Andreas Gruenbacher
  - We already have info levels mapped to get/set POSIX ACLs.
- Linux may be adding RichACLs (Andreas Gruenbacher's code)
  - Do we map these into Windows ACLs, or create new info levels ?

Implementing in Samba: The ACL Solution ?

- Microsoft has a well-kown mapping for POSIX permissions mapped into Windows ACLs for their NFS implementation.
  - Get/Set ACL on Unix Extension handle could return “best effort” mapping of what is on disk to the client.
  - Leave it to the client to sort out what that mapping means locally.
  - Given that this mapping is well-known this would allow local Linux RichACL mapping to server.
  - Does this cover the ACL mask ?

Implementing the SMB3 UNIX Extensions in Samba



- Prototyping will be done by adding calls to smbclient (the cli_XXXX() internal Samba library) to exercise new features in the server.
- Feature set and behavior must be agreed upon with the Linux CIFSFS client implementors.
  - Avoid SMB1 UNIX extensions mistakes like the encryption support.
- Eventually expose to libsmbclient library used by Gnome applications like Nautilus (file browser).
  - Make available to Gnome VFS users.
- What about the BSD-of-the-month club and Solaris ?

Opening Windows to a Wider World

The Devil in the Details

- Interactions between different handles will have to be explicitly defined and documented.
- Recent issue – delete on a Unix extensions handle containing named streams previously added by a Windows client caused the server to crash.
- How much of these are Samba “implementations behavior” notes, how many are protocol-relevent details ?

The Definition of Success: Windows Server support ?

- Long term, to support Linux clients in a Windows cloud file server, Microsoft may end up needing to support SMB3 UNIX extensions.
  - This will be dependent on market demand for Linux clients in a Windows cloud.
  - Microsoft Azure SMB3 file server might be easiest target here as it's a new implementation.
- It's worth spending time getting the design right to make this possible.
  - Don't repeat mistakes of SMB1 UNIX extensions

Questions and Comments ?

Email: jra@samba.org

Slides available at: <tbd>