Linux Kernel Clients: A Year in Review
“Man-Eating Seals of Antiquity”
to
“Sheep on Meth”

Steve French
IBM – Linux Technology Center
Legal Statement

– This work represents the views of the author(s) and does not necessarily reflect the views of IBM Corporation.
– A full list of U.S. trademarks owned by IBM may be found at http://www.ibm.com/legal/copytrade.shtml.
– Linux is a registered trademark of Linus Torvalds.
– Other company, product, and service names may be trademarks or service marks of others.
Who am I?

– Steve French (smfrench@gmail.com or sfrench@us.ibm.com)
– Author and maintainer of Linux cifs vfs (for accessing Samba, Windows and various SMB/CIFS based NAS appliances)
– Wrote initial SMB2 kernel client prototype
– Member of the Samba team, coauthor of CIFS Technical Reference and former SNIA CIFS Working Group chair
– Architect: Filesystems/NFS/Samba for IBM LTC
• Last September Kernel version 2.6.31:
  • “Man-Eating Seals of Antiquity”
    – 2.6.31 came out on September 9th, 2009

• [Since 2.6.34] Now we have 2.6.36-rc4

• “Sheep on Meth”
Development on Linux kernel clients is very active

- 187 kernel changesets for cifs since 2.6.31 (typical year)
- But … improvements to related tools have accelerated dramatically: in a few month period early this year 100+ changesets in mount helper and related utilities were merged
- And smb2 development accelerated …
  - A month after SambaXP last year, public git tree for smb2 development created on kernel.org
  - More than 130 kernel changesets for smb2 since January
  - SMB2 module now close to full function but 30% smaller (17188 lines of code, CIFS is now 24447 LOC). SMB2 should shrink 10-20% as common code moved to helpers.
  - SMB2 has few dialects, no redundant commands, no buggy backlevel servers to workaround (yet), and name handling much simpler (UCS-16 only), and 32 bit status errors only)
Just in case you forgot the goals ...

- Local/Remote Transparency
  - Most applications shouldn't notice or care if on remote mount vs. ext4
- Near perfect POSIX semantics to Samba servers (and those which implement POSIX extensions) and best effort semantics to Windows and other NAS filers
- Fast, efficient, full function gateway for accessing data from Linux which lives on Windows & NAS
- As reliable as reasonably possible over bad networks
- But what about SMB2? The protocol offers many improvements which will benefit Linux:
  - Add scalability that cifs can not handle (e.g. increased number of open files, more inodes)
  - Higher performance on fast networks:
    - very large i/o, flow control (credits)
  - And also slower networks
    - Better caching
    - Better operation compounding
  - Offer stricter data integrity guarantees than cifs can offer
The Team (More than 20 developers contributed)

- **CIFS** (both Jeff and Steve were in top 20 of overall Linux fs contributors again this year). Among the contributors to cifs this year were:
  - Jeff Layton
  - Steve French
  - Shirish Pargaonkar, Pavel, Suresh
  - And many more

- **SMB2**
  - Steve French
  - Pavel
  - Shirish Pargaonkar
  - Jeremy Bongio
  - And more
CIFS
Celebrating 7 years in the Linux Kernel
Busy year

- Now cifs.ko version 1.65 (Linux kernel 2.6.34-rc6)
- A year ago was CIFS version 1.58 (Linux kernel 2.6.31)
- Two years earlier huge amounts of Microsoft WSPP documentation released
  - Recently MS-CIFS released on msdn which may help with identifying corner cases which are hard to handle
CIFS Progress

- Open bugs in bugzilla down to about 40 (including some which are now unrepeatable or invalid or feature requests)
  - Stability improvements:
    - Multiple buffer overruns fixed, ntlmv2 memory leak fixed
    - Unlink and symlink improvements
    - DFS loop checks and DFS error handling
    - POSIX open/create to various differing Samba servers fixes
    - Reconnect fixes, append fixes, timestamp fix
    - Byte range lock fixes (including losing locks on fork)
    - Fix large writes to OS/2 servers
CIFS Progress (continued)

- “Raw NTLMSSP” (auth) support fixed and reenabled (sec=ntlmssp)
- Larger symlinks supported (from 4000 to PATH_MAX)
- O_SYNC fixes
- Support for longer (>35 characters) IPv6 addresses
- /proc/mounts now shows more cifs info, and cifs stats improved
- Tighter default posix mode
- Oplock and cache validation improvements
- Openoffice fixed over forcedirectio mounts
- Locking, serialization improvements (bkl)
- Using server generated inode number now the default in most cases (mount option “serverino”). This may change back temporarily in 2.6.34 due to a bug
- Cifs over ssh tunnel fixed, cifs for network boot fixed (diskless boot oops), cifs supports leases (getting and breaking leases)
- Make default uid/gid behavior more intuitive and flexible: mounts by unpriv users when Unix Extensions on can now show real (server generated) uid/gid, noforceuid and noforcegid mount options added so that ownership can always be preserved if needed
- TCP_NODELAY sock option can be sent (perf improvement)
CIFS MultiUser Support and Improved Authentication

- NFS sends (at least) primitive Unix credentials via SunRPC but cifs uses default credentials (supplied at mount time) for all users
- CIFS had a “MultiUserMount” feature which was awkward to use
- Jeff Layton has been adding a much improved MultiUser Support for CIFS (see his talk from SambaXP)
  - Merging now into 2.6.37-pre cifs-2.6.git tree
- (Following 4 slides courtesy of Jeff Layton)
What happens on file creation?

- File was created on server using mount credentials
- CIFS attempts to enforce permissions on client
- That can't fix ownership
- File is created but later ops fail!
Why is it this way?

- CIFS protocol is **session-based**
- Credentials are handled per-session
- Linux CIFS only has single session per mount
- **Shared Credentials!**
The solution...

- Each user should use their own credentials
- Have multiple sessions per mount
- Establish sessions on an as-needed basis
- Let the server handle permissions
- **Goal:** Easy as NFS
The CIFS protocol has a hierarchy of sorts

- NEGOTIATE
- SESSION_SETUP
- TREE_CONNECT

- Open filehandles
- Other path-based ops
Basic Object Hierarchy

- TCP_Server_Info (socket)
  - per socket
  - NEGOTIATE
- cifsSesInfo (credential)
  - SESSION_SETUP
- cifsTconInfo (share)
  - TREE_CONNECT
- cifsFileInfo (file)
  - open filehandles
CIFS User Space Utilities

- Split into new git tree (cifs-utils) to ease maintainability in February (big success – lots of fixes)
  - Fixes to cifs.upcall (for DNS resolution and kerberos authentication support, getting krb5 tickets and handling spnego)
  - Fixes to mount.cifs (mount helper)
    - Much easier to build and more stable
    - Mounts by less privileged users now safer
  - Since SambaXP have mount.smb2 helper (sharing code with mount.cifs.c)
FS Cache

- Oplock allows client caching. Linux has a page cache mechanism which is heavily utilized by cifs
- FS Cache allows cifs to use Dave Howell's file system caching mechanism to cache files to disk.
- Especially useful on slow networks or very heavily loaded servers, where reading from disk is faster than reading from the server
Network Interface Selection

- Can now select which network interface (e.g. wireless vs. wired) that a cifs mount (tcp connection request) is sent over (Ben Greear)
- Merged into cifs-2.6.git (will go into 2.6.37)
- Similar patch in smb2
New Symlink Support (Metze)

- Especially for servers which don't support Unix Extensions, need alternatives for creating symlinks
  - SFU/SUA style is ok
  - On recent Windows servers SMB2 style symlinks may also be possible
- Now have patches to support MacOS style symlinks
  - Expected to go into 2.6.37
  - Will allow creating/reading symlinks to servers, like Windows, which don't support the Unix Extensions
  - This type of symlink is not typically read by the server (e.g. Windows) so is relatively safer to create than those which can be followed by server side applications (since the client can't follow these to server paths outside of the share)
Authentication Improvements

- Caching uid information using the Kernel Credential Keyring
  - Another Google Summer of Code project
  - User space improvements merged
  - Waiting on fixes to kernel patches (possible for 2.6.37)

- Shirish's auth improvements
  - With these patches (raw) NTLMv2 authentication now fixed to Windows 7 and Windows Vista
    - More recent Windows is pickier than Windows XP and Windows 2003 about format of NTLMv2
  - Fixes NTLMv2 in NTLMSSP as well
  - CIFS now uses kernel crypto routines (e.g. for MD4 and MD5) instead of cifs specific implementation, reducing code/module size
SMB2
Network File System of the Future
History

- Prototype begun about 2.5 years ago then rewritten a year later
- Pavel Google Summer of Code helped make some progress on this in summer 2009. Mount working reasonably by end of summer
- Found various spec problems and bugs at MS test events
- Work restarted in January with Jeremy Bongio helping (improved read support)
- Pavel back on Google Summer of code this summer – wrote fast async write
- Git tree created on kernel.org at the end of May 2009, based on rewrite of early prototype
- Tested at 2009 Microsoft plugfest (focus on mount and authentication)
- Tested at 2010 Connectathon (many file operations working, added hardlinks and rename and jra fixed some server bugs in those areas too)
- Currently at version 0.48
Status

- 17188 LOC (much smaller than cifs, partly because SMB2 is much simpler, fewer redundant operations, handle based, and also partly due to cleanup)
- Working operations:
  - Mount, revalidate, read, direct i/o write, create, delete (unlink), mkdir, rmdir,
  - Hardlink, rename, readdir, readpage (Jeremy), byte range locks (Pavel)
  - Async writepages (Pavel)
- Work in progress
  - Readdir fixes
  - Oplock improvements (Steve)
  - Async readpages
  - NTLMSSP and auth improvements (Shirish)
- Key missing operations:
  - Symlink,, setattr acl/mode operations
- Goal to merge into linux-next within next month (at least as “developer” or “experimental”) assuming testing goes well here this week
Preliminary Performance Results

• Generally SMB2 benefits from three factors
  – Larger i/o sizes
  – credit based flow control (easier to achieve more parallelism)
  – Improved caching model

• email from Jeremy Bongio (February)
  – To server on local vm (cat remote-file > /dev/null) current Linux kernel clients to Samba server
  – SMB2 readpages about 20% faster already for medium to large file read (should be relatively even faster over physical network)
  – smb2
    – 166K 0.0342 sec
    – 200000K 11.012 sec
  – CIFS
    – 166K 0.0476 sec
    – 200000K 13.230 sec

• Pavel's async write code
  – For writing medium/large files to Linux/Samba server, performance ~30% better than NFS (Linux to Linux) and more dramatic compared to cifs
Performance results continued

- (Quoting Pavel's post last month)
  - SMB2 can be faster than NFS and CIFS on Linux. I got the following results after $10^5$ writing each of 4096 bytes of data.

- SMB2 : ~0m 29s
- NFS    : ~0m 40s
- CIFS   : ~1m 28s

The test environment:
- Client    - Ubuntu 9.10 with 2.6.33 kernel on VirtualBox.
- Server    - Fedora 12 with 2.6.32 kernel, Samba 4.0.0alpha11.

Machines were connected through bridge.

test is rather synthetic but it shows at least the one side of SMB2 advantages.
Performance Results (Cont) – SMB2 twice as fast

- This morning to jra's most current Samba 3.0.6 server from current Linux smb2 and cifs kernel clients
  - time dd if=/dev/zero of=/smb2mnt/file1 bs=1M count=100
  - 100+0 records in
  - 100+0 records out
  - 104857600 bytes (105 MB) copied, 1.31032 s, 80.0 MB/s
  - time dd if=/dev/zero of=/cifsmnt/file2 bs=1M count=100
  - 100+0 records in
  - 100+0 records out
  - 104857600 bytes (105 MB) copied, 2.64377 s, 39.7 MB/s
What is SMB2?

- Default filesystem protocol for Windows (since 2008).
  - Samba must support it well as a server
  - Current server implementation functional, but layered over SMB/CIFS rather than optimized, missing full implementation of some new protocol features

- Rapidly becoming most common network file system protocol
  - To get data most efficiently from other systems (not just Windows but most NAS), we must have great support for SMB2 kernel client (not just server) for Linux as well
  - To make Linux (and other Unix) even better (to Samba e.g.) adding minor extensions to Linux client/server
  - SMB2 better than CIFS for use by Linux for getting files from NAS

- What is it?
  - SMB2 is not simply a new dialect of SMB/CIFS
  - Protocol features continue to improve (already have an SMB2.1)
  - SMB2 enables new performance, security and reliability features
    - larger i/o sizes, caching improvements, credits, compounding, async operations, scalability greatly increased etc.
    - Features have synergy with Samba/Linux's strengths
SMB2 Implementation Design Goals

- Faster than CIFS
- Improve Samba server through cooperative testing
- Cleanup many of the small design and code problems noticed after coding cifs
- Experiment with features that are too risky to do in stable cifs
- Allow Higher Data Integrity guarantees through use of the new SMB2 protocol features in this area
- Set default security settings to higher level than would be possible with cifs (which supports many older, buggy servers)
- Testbed for Unix Extensions
SMB2 Kernel Client Priorities & Plans

- **Basic compatibility**: Enough function so Linux kernel client passes at least as many functional tests as Linux CIFS client (to Windows)
- **Basic compatibility**: Samba server passes basic functional tests (from Linux or Windows SMB2 clients)
- **Optimizations for Samba**:
  - Define “POSIX SMB2 Protocol Extensions”
    - Reserve command codes, information levels (with Microsoft)
    - Get agreement with Samba SMB2 server developers on extensions
    - Extend VFS and/or protocol in order to implement ALL local fs interfaces over SMB2: Maximize app compatibility
  - Ensure SMB2 is faster than CIFS to Samba/Linux
- **Community acceptance**:
  - Expand community
  - Redesign problematic, hard to maintain areas in CIFS kernel client;
  - Add SMB2 kernel client to linux-next then mainline (experimental)
  - Aiming for mainline for 2.6.36 (approximate)
  - Test events like this (and the summer plugfest in Redmond) help.
Unix Extensions Redux: SMB2
Purpose of Unix Extensions

- Provide exact file semantics from Linux, Solaris, MacOS and other POSIX clients
  - Real world compatibility requires few operations outside of the posix spec (getattr for example)
- A network file system must provide transparency – look like a local file system, and not break common applications
- Compensations for non-POSIX style operations must not harm data integrity, and have only limited impact on performance
- Where reasonable Unix Extensions should be designed so the could be implementable by Windows and others non-Unix/Linux servers
Some History

- Recently Microsoft have begun work with community on extensions
  - Web site now set up
- Earlier proposal at fall SNIA 2008 conference on a mechanism using existing SMB2 open extensions to pass Unix specific data was well received
  - Utilize Create Context
  - Allow posix behavior to be specified on an opens
    - Would allow others apps on same client to open file with windows semantics (using different handle)
    - Does not address all Unix requirements, but helpful
- See Tom Talpey's talk on Wednesday on SMB2 Unix/POSIX Extensions!
Key Areas to discuss

- Is remapping (0xF000 + char) good enough for handling reserved characters which posix needs (including “:” and “*” for example)
- Can we request case sensitive searches?
- How to request posix semantics on unlink and rename (e.g. for open files)
- POSIX byte range locks (advisory)
- Returning mode bits (is emulating via SMB2 ACL enough)
- UID/GID owners (require client to resolve UUIDs in SID to local uid UID?)
- Are UniqueIDs really unique enough to be inode numbers (do we have device identifier and unique file identifier that we can use to properly hash dentries and inodes on the clients without collisions)?
- Are UniqueIDs persistent after reconnection (very hard to flush all cached dentries and inode metadata on client every time network connection drops)
- POSIX ACLs (Linux’s default ACL model)
- Getattr (to get at inode metadata like “immutable” flag and append only flags)
- NFS export of SMB2 (NFS export operations to resolve NFS file handles)
- How to detect what type of ACLs are preferred by the server file system(s)?
- Useful general features (transport encryption) supported in cifs unix extensions
Thank you for your time!!