The Future of File Protocols: ~~CIFS~~ ~~*SMB2*~~ **SMB3** Meets Linux

Steve French
Senior Engineer – File System Architect
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 SNIA CIFS Technical Reference and former SNIA CIFS Working Group chair
- Architect: File Systems/NFS/Samba for IBM Linux Technology Center

SMB2: Great Feature Set, Broad Deployment, Amazing Performance

- Introduction of new storage features in Windows 8 causing one of most dramatic shifts in storage industry as companies rapidly move to support “SMB3” (aka SMB.2.2) file protocol
- “SMB2.2 (CIFS) screams over InfiniBand” (Storage CH Blog)
- Is (traditional) SAN use going to die?
  - “Market trends show virtualization workloads moving to NAS” (Dennis Chapman, Technical Director NetApp, SNIA SDC 2011)
  - “Unstructured data (file-based) storage is growing faster than structured data” and IDC data shows customers prefer network file storage (Thomas Pfenning, Microsoft GM, SNIA SDC 2011)
- SMB2/CIFS market share is MUCH larger than NFS. pNFS/NFSv4.1 deployment slow

Performance Caught My Attention too … SMB3.0 is FAST

- At the SNIA Developer Conference last fall... an amazing demo. SMB2.2 using RDMA over InfiniBand reached over 3.7GB/sec throughput for a common database benchmark with no server configuration changes using only two cards. 160K IOPs (database SQLIO run). Pre-beta SMB2.2 code running on commodity server hardware using 32Gbps InfiniBand links (see http://www.mellanox.com/content/pages.php?pg=press_release_item&rec_id=735)
- Similar performance (albeit with higher CPU on the client) was achieved on the same systems, when 4 commodity 10Gb Ethernet cards used instead
- It gets better … Microsoft demonstrated even faster performance with the recent beta (and yesterday at SambaXP more was posted!)
- Storage Developers asking is "Network" (SMB2 File Access) now better than "Local"?

- And it looked EASY to setup!

SMB/CIFS and SMB2 – A Little History and Terminology

- “SMB” (later called “CIFS” - “Common Internet File System”)
  - Originally created by IBM for PC-DOS (Dr. Barry Feigenbaum et al) 1984
    - Documented by IBM
  - Updated extensively for OS/2 by IBM and Microsoft (1988-1992)
    - Standards Document by X/Open (OpenGroup). Updated again in 1992 “Protocols for X/Open PC Interworking: SMB, Version 2” (not to be confused with SMB2!)
  - Microsoft continued to extend the protocol, and include it in Windows clients and servers (but without public documentation)
  - Renamed to “CIFS” in 1997 which was documented by Microsoft and SNIA
- SMB2
  - Documented in unprecedented detail (for a network file system protocol) since 2007 (through WSPP), with test cases to prove docs right, and network analysis tools
  - Introduced with Windows Vista (and Windows Server 2008)
  - V2.1 with Windows 7, W2K8R2 Server
  - V3 will come in Windows 8 (and various competitor's NAS products)
- SMB2 is a “De facto” not an “open” standard (cifs implemented broadly including Mac, Solaris, Linux). CIFS or SMB2 default on most shipping PCs, and is the most popular network file protocol (by far).

Who Works on CIFS/SMB2? The Amazing Samba Team

Samba: User Space Server … but Linux client is in kernel (and only loosely related to Samba)

- Picture shows many of the members of the Samba Team as they gathered last year at the 10th annual Samba eXPerience conference in Göttingen, Germany.
- There are Kernel Developers too:
  - Me
  - Jeff Layton (RedHat) maintains cifs-utils
  - Pavel Shilovsky (Etersoft)
  - Engineers from RedHat, SuSE, IBM, GSoC and other companies help out too
- Samba team focuses on File Server, tools, Winbind (authentication helpers), and Active Directory Domain Controller
- Linux Kernel Developers (three of which are also members of the Samba team) focus on the high performance network file system kernel client (to Samba, Windows, and other NAS servers) and related tools

Although network API closer to Windows than POSIX, CIFS and SMB2 not really Windows specific

- Mac, Solaris, Linux and most other operating systems have kernel clients. Solaris and Mac even use CIFS ACLs in-kernel. CIFS/SMB2 default for some Unix and all Windows.
- “Unix Extensions” developed by SCO, extended by HP and then Linux and Mac. Improve most “posix vs. windows” issues such as retrieving the Linux ACL and POSIX locking.
- CIFS Unix Extensions implemented in Samba and Linux kernel client among others.
- Unix Extensions are optional (when mounted to Windows, they are emulated instead, sometimes using the same approach as “Services for Unix”). Mount from Linux to Windows just works for most applications. NB: NFSv3 is not completely POSIX friendly
- Microsoft made SMB2 slightly more “unix friendly” so extensions for SMB2 will be smaller

SAMBA
opening windows to a wider world

What about NFS?

- Once many distributed/network file systems, now down to two popular ones:
  - NFS
  - CIFS/SMB2
- And some special purpose cluster specific fs (note that you couldn't run general applications on http and Hadoop mounts so those aren't considered network file systems)
- NFS
  - Created by Sun 1984 (roughly the same time as SMB) who documented NFSv2 in 1989, and documented NFSv3 in 1995,
  - NFSv4 became an open internet standard relatively late: RFC 3530 in 2003. It heavily borrowed from cifs features (oplock, ACLs, DFS referrals)
  - An enormous update (more than 600 pages), NFSv4.1 became a standard in 2010, and included pNFS support and better cifs/windows ACL interoperability
  - But NFS v4.1 Implementations have lagged, and even NFSv4 deployment slower than expected. Most Linux NFS users still use 17 year old NFS version 3, even though it is not fully posix compliant and can not do safe caching
  - Work continues on an opensource userpsace Ganesha NFS server with NFSv4.1/pNFS support, and also the kernel pNFS kernel server (currently maintained out of kernel). Few pNFS clients other than Linux exist, and RHEL 6.2 only very recently added limited support for pNFS client.

SMB vs. NFS

- SMB1
  - Stateful
  - Per-user connections
  - Directly uses TCP (RFC1001)
  - Can be used as a transport for other protocols (DCE/RPC, print, systems management)
  - Originally optimized for DOS/OS2 then Windows
  - Rich: Lots of file operations

- NFSv3
  - Stateless (mostly). Idempotence allows operations to be repeated
  - Does not guarantee safe caching
  - Per-system connections
  - Runs over SunRPC transport protocol
  - Originally for Solaris
  - Minimal: Few file operations, almost a “lowest common denominator” across Unix

Current Versions (SMB3.0 vs. NFSv4.1)

- Both have borrowed from each other: NFSv4 in particular added various cifs features (including statefulness, and various security features)
- SMB3.0 and NFSv4.1 both include:
  - Kerberos authentication, packet signing, encryption
  - “RichACL” (CIFS ACLs)
  - Support for file transfers via RDMA
- NFSv4.1 includes optional pNFS (file or block or object) to spread network i/o load from a single client across a cluster
- But SMB3.0 and related protocols now include
  - Multipath, per-share encryption, better server side copy, support for copy on write files, claims based access control, branch caching (content addressable storage), volume shadow copy, improved cluster awareness and load balancing, T10 extensions, flow control on every response, application aware and also transparent failover

The Dinosaurs, created in the same Orwellian year are reborn – faster and stronger than ever

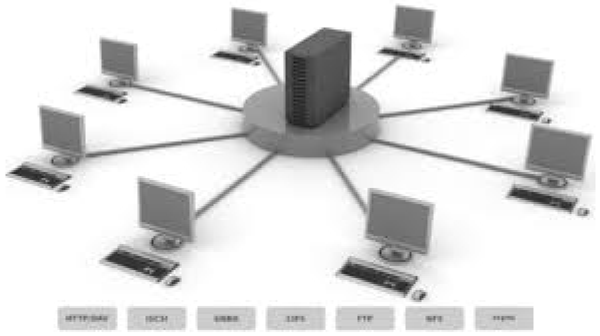
Will NFSv4.2 Address SMB 3 gaps?

- NFSv4.2 specification does include some items to close gaps:
  - Server side file copy
  - “punch hole” support
  - Fadvise (indicate file access patterns) and allocate (space reservation) support
- And bug fixes (for NFSv4/NFSv4.1 spec problems)
- Fortunately 4.2 is a much smaller update than NFSv4.1 (1/7th the document size).
- But … SMB 3 will have MUCH wider deployment, and others in NAS community (EMC and NetApp) already indicated support forthcoming for SMB3
- NFSv4.2 implementation likely to lag (a lot) behind SMB 3 (NB: still working on NFSv4.1)
- On the other hand … SMB 3 Unix Extensions are not complete yet so there are a few gaps for SMB 3 to close
- Although not likely to be as widely used outside of Linux as SMB 3, the amount of NFS4.2 usage in pure Linux (ie Linux client mounts to Linux server) will depend in part on how high quality the SMB 3 Linux implementations in kernel and Samba server are.

What about other Network, Cluster, Distributed, Cloud … File Systems?

- Cluster File Systems (GPFS, Luster, GFS2, OCFS2)
  - Narrow target market
    - Can run well on either one or only a few operating systems
    - Support not included with shipping PCs (Windows and Mac)
  - They will continue to have a place in the server room “behind” the servers for the more common protocols (SMB2, NFS, and in some cases Web).
  - With SMB 3 and pNFS so fast - don't mount clusterfs directly (let Samba srv do it)
  - They will continue to have a place in HPC, but more limited due to SMB2 and pNFS
    - NB: “Watson” did use GPFS, but behind NFS servers that HPC nodes mounted
- OpenAFS, GlusterFS, “CloudFS”
  - Performance issues (not just due to usual problems with user space file systems)
  - Narrow target market
    - Can run well on either one or only a few operating systems
    - Support not included with shipping PCs (Windows and Mac)
  - They don't tie into enterprise archive, data retention, security tooling on NAS
- Virtfs
  - Great for Linux guest to Linux host mounts (on the same box) due to virtio (could do virtio for SMB2 if desired) and strong Linux affinity
- For the vast majority of use cases – it comes down to NFS or CIFS/SMB2 (now SMB 3)

What about using block devices instead of file?



- SAN vs. NAS, block vs. network file access
- Why use file? Well … it is cheaper AND
  - Its easier to administer and setup: Users and Admins understand them
  - Security is easier on files (which have owners and ACLs)
  - Setting up Data Retention (Backup) and dedup correctly is easier on files
  - Application Access patterns can be optimized better on file
  - More flexible (block devices can be emulated as files on NAS mounts)
- Why use block?
  - Faster (?!) - is that still true?
  - Redirectors and file servers had implementation problems (inode locking, aio) which hindered scaling, especially with virtualization and database
- But what if Network File (with SMB 3) were now as fast as block ...? Would SAN move to more of a niche role, sitting behind the cluster fs on high end NAS filers?

SMB2 is Everywhere

- Quoting Dan Shearer: *“The costs and risks of having a non-default filesystem for Windows are rarely worth small performance advantages (at best.)”*
- A monopoly in operating system market gave us a ubiquitous network file protocol (SMB2)
  - Well documented
  - Well tested
  - Frequent interoperability test events
- Protocol broadly adopted by others
- SMB 3 will be VERY widely deployed within a year, and keep growing as NAS and other OS implement it.
- But Linux has MANY advantages to allow us to implement SMB 3 efficiently

CIFS/SMB2 is not just a file protocol – it addresses complete picture

- When comparing SMB2 to NFS or to another cluster file system remember the big picture
  - It is not just about files, but how to find them and manage them
  - SMB2 easily wins over alternatives when looking at easily managing, securing data
  - In IBM we run Samba/GPFS in GSA (our server farm). Easy and clients work out of the box
- SMB2 (and SMB 3) are integrated with
  - “DFS” (global namespace)
  - Advanced security features (not just Kerberos authentication)
  - A rich management model (DCE-RPC, LDAP and CIM based) which uses the same access control and authentication. Samba can handle thousands of types of administrative requests remotely to manage everything from desktop settings, to DNS, to file server exports via PowerShell and MMC (Windows) or use Linux's “net” and “samba-tool” and smbclient on Linux, or use the many third party systems management tools
- Domain Controller (central security and configuration server) ties resource configuration and security together in one place across multiple protocols (Kerberos, LDAP, DNS, DHCP, NTP, DCE/RPC, Print, SMB2)
  - Linux implementation based on Samba 4 looks great (check back at SambaXP this May for an update)
- Generally it is easier to manage SMB2 because the many existing tools just work, whether to Windows, or NAS, or Samba. Very rich management/admin environment

SMB2.2 Summary

- **SMB 3 will be critical to Linux, and important to our customers**
  - Goes beyond traditional CIFS/SMB2 strengths (already the most popular file protocol) and addresses key enterprise workload requirements for virtualization, HPC, Database and Web
  - Adds significant performance and functional improvements. In many cases SMB 3 will be as fast as raw access to network block devices
- **At Windows 8 launch, wide deployment expected to begin**
  - At SDC NetApp and EMC announced forthcoming SMB2.2 support
  - SMB 3 will have an impact on NFS
- **Part of a broader shift from "block" to "file" (even for high end workloads)**
  - File is cheaper and easier to setup. SMB 3 addresses some of the remaining performance gaps making the transition even more compelling.
- **Samba community committed to delivering base level support in server (client support likely too). Due to breadth of workloads addressed by protocol, and complexity of new features will need help to achieve our goals**
- **Key SMB 3 features play well to Linux/Samba strengths on high end hardware**
  - Cluster friendly (cross-cluster shares, failover "witness protocol", better HA features ...)
  - Larger i/o sizes
  - More parallel access, clients are more scalable
  - Metadata caching
  - Optional RDMA, and T10 enablement

SMB 3 – what does protocol improve on?

- **Addresses requirements of 4 key enterprise workloads and Improves**
  - Availability
    - Enable transparent client recover in the presence of Network Failure
    - Server Failure
    - Minimize failover time to reduce application stalls
    - Also allow planned, application initiated failover
  - Performance
    - Enable clients to aggregate available bandwidth across adapters transparently
    - Continue to increase efficiency on high bandwidth networks
    - Cluster enabled to allow for higher scalability
  - Traffic Reduction
    - Continue improving user perceived latency when working in a WAN environment
- **Key features:**
  - Multichannel
  - SMB over RDMA
  - Scale-Out Awareness
  - Per-share encryption (and security even better in other areas too ...)
  - Persistent Handles
  - Witness Notification Protocol
  - Clustered Client Failover
  - Directory Leasing and improved metadata caching
  - Branch Cache v2 (content addressable storage)
  - Support for Storage Features (TRIM, block size discovery, T10 etc)
  - Claims Based Access Control

Development on Linux kernel clients and Samba is very active

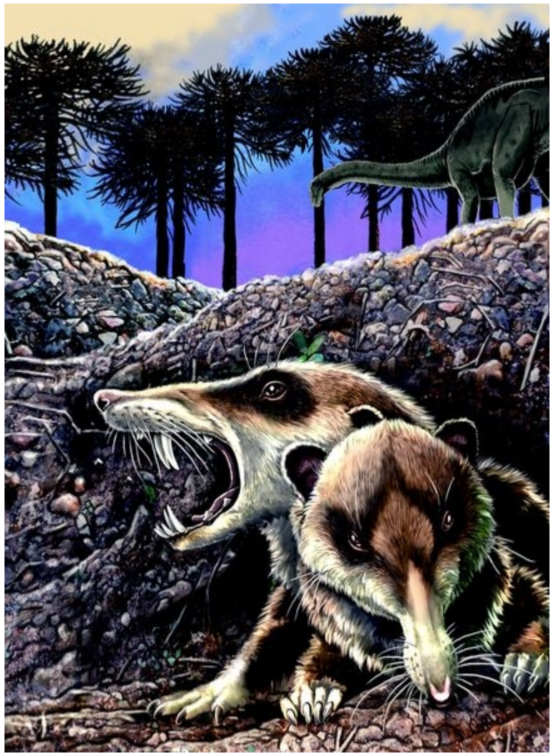
- Kernel client (cifs.ko)
  - Current version is 1.78 (visible via modinfo)
  - 338 kernel changesets for cifs since 2.6.38, a very active year
  - More than 20 developers contributed
  - cifs continues to be one of the more active file systems
  - And … improvements to related tools have accelerated dramatically:
  - Development expected to grow even more when SMB2 support merged
- Samba server
  - Current stable version is 3.6.3
  - Current development version is 4.0 (nearing beta)
  - A year ago Samba 3.5.8 was current
  - Samba grown to over 2 million (!) lines of code, including Active Directory Domain Controller, Management, Security features, and an amazing file server and tools

Samba SMB 2 status

- **SMB2.0 status: server passes all functional tests (included in Samba 3.6)**
  - Simplified command set and uniformity (UNICODE, timestamps, etc.): Complete
  - Expanded identifier space (UINT64): complete but emulation awkward in some places
  - HMAC-SHA256 signing: complete
  - Dynamic crediting: partial implementation (not optimized to retrieve from FS)
  - Async notifications for long running requests: complete (not GPFS aware)
  - Unrestricted compounding of requests: complete
  - Symbolic Link support: not complete
  - Durable opens for handling disconnect: emulated, needs work (VFS dependency)
- **SMB 2.1 status (still early for server, but kernel client slightly farther)**
  - Frame reduction for common workloads and WAN
    - SMB Leasing, critical performance feature missing, server not complete (VFS dependency, but can be emulated in server in the near term), but ok in kernel client
    - Branch Cache extensions: prototype only (outside of Samba, in python)
  - Large MTU support (throughput)
  - Resilient Handles: not complete (VFS dependency)
- **SMB 3 status: work begun in late September in server (work below included in forthcoming Samba 4, can be downloaded from samba.org)**
  - Client Failover from one Samba server to another (e.g. during DVD copy)
  - SMB2.2 packet signing complete (AES-CMAC)
  - “Persistent” and “Durable” handles emulated
  - Next test event this summer
  - Goal to have implementation by Windows 8 launch
  - See http://blog.obnox.de/samba-team-visits-microsoft-for-smb2-2-interop-event/

Kernel (including the cifs client) improving rapidly

- Current Kernel (3.4-rc6) “Saber-toothed Squirrel” (Cronopio dentiacutus)
  - http://news.nationalgeographic.com/news/2011/10/101102-saber-toothed-squirrel-fossils-paleontology-dinosaurs-science/



- A year ago we had 2.6.39-rc2
- “Flesh-eating Bats with Fangs” (name since 2.6.36-rc8)

CIFS Improvements (Linux Kernel Client)

- Performance improvements
  - Async write
  - Async read
  - Faster vectored AIO and forcedirectio (write in 3.4, read 3.5)
  - Readahead to Samba faster (3.4)
  - Maximum number of requests to one server increased from 256 to 32K
    - Most servers default to 50 simultaneous requests. requires update to smb.conf (Samba) or Windows server registry
- CIFS is VERY fast now on Linux to Linux (not just to Windows)
  - Great way to transfer large files
  - Not just the client has improved, Samba server performance also faster now due to improved dispatch of reads and writes as well

- Security and other functional improvements
  - Multiuser mounts using NTLMv2 authentication (easier to setup than Kerberos, for which cifs already had multiuser support)
  - SetACL (can set CIFS ACLs via xattr)
  - Get/Set SID (Windows Owner)
  - Enterprise backup improvements
    - Including "BACKUP_INTENTS" (backup operator support)
  - FS Cache support (offline cache)
  - Samba tcon encryption (nearing completion, obstacle was how to demultiplex multiple secure mounts with different users on one socket)

Some recent perf improvements

- cifs_iovec_read now collects/issues (larger) asynchronous reads. Primarily of use when a share is mounted with forcedirectio, or strictcache and the client doesn't have an oplock for the file being (Will be in 3.5. From Jeff Layton)
- Big increase read performance there
- Test results from my low end KVM test rig to samba. Running simply:
  - `$ dd if=./ddtest.out of=/dev/null bs=1M`
- Results:
  - Unpatched 3.4-rc2 kernel -- rsize is always capped at 16k here:
    - 1073741824 bytes (1.1 GB) copied, 97.6394 s, 11.0 MB/s
  - Patched 3.4-rc2 kernel – rsize=1M:
    - 1073741824 bytes (1.1 GB) copied, 9.89869 s, 108 MB/s
  - Patched 3.4-rc2 – rsize=61440:
    - 1073741824 bytes (1.1 GB) copied,   13.4146 s, 80.0 MB/s

Other recent perf improvement (thank you Jeff Layton)

- Normal buffered large file read got MUCH faster on 3.2
  - 1GB file copy from server to /dev/null with dd (on slow kvm test system, would be more dramatic improvement over fast network to fast server)
  - prepatch, with 16k rsize:
    - 1073741824 bytes (1.1 GB) copied, 47.2119 s, 22.7 MB/s
  - postpatch, with 1M rsize:
    - 1073741824 bytes (1.1 GB) copied, 11.1602 s, 96.2 MB/s
  - postpatch, with 60k rsize:
    - 1073741824 bytes (1.1 GB) copied, 12.5183 s, 85.8 MB/s

- Readahead had been often limited to 128K due to kernel bdi defaults – but when mounted to Samba which supports MUCH larger reads this limits perf gain of readahead
  - Patch to improve this in 3.4 kernel (slow kvm guest mount to local Samba on host – see below)
  - $ dd if=./ddtest.out of=/dev/null bs=1M count=1024
  - Prepatch:
    - 1048576000 bytes (1.0 GB) copied, 28.1979 s, 37.2 MB/s
  - Postpatch:
    - 1048576000 bytes (1.0 GB) copied, 11.92 s, 88.0 MB/s

Even with SMB2.1 Kernel Prototype: We were already faster (GsoC test results, August last year)

	CIFS	SMB2	SMB2.1
1 fd for reading	20.5s	21.5s	3.82s
2 fds for reading	32.7s	31.5s	3.96s
3 fds for reading	41.6s	40.6s	4.08s
4 fds for reading	51.3s	51.5s	4.2s
5 fds for reading	61.8s	64.4s	4.33s

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

SMB2 Kernel Client Status

- SMB2 kernel code is experimental, partially merged but getting closer to ready due to some good work by Pavel
  - supports SMB2 and SMB2.1 dialects (defaults to SMB2.1)
  - Passes most “connectathon” functional tests (turning on SMB2.2 caused it to fail one additional test)
- Decision made last year to use a common cifs.ko module for BOTH cifs and smb2 support for Linux (Microsoft uses separate redirector drivers for each for Windows)
- Common code turned out to be larger than expected, but still leaves about 6,000-8,000 lines of SMB2 specific code
- Core cifs code has now been modified to extend fields in preparation for SMB2 merge (complete except for byte range locking changes, which are being reviewed)
- SMB2 support in distinct .C files (optionally configurable if desired to be built) to reduce chance of destabilizing cifs code
- Pavel's git tree with cifs.ko updated with latest smb2 kernel client code:
  - git.altlinux.org/people/piastry/public/?p=cifs-2.6.git;a=shortlog;h=refs/heads/smb2-dev

SMB2 Kernel Client Plans

- SMB2 expected to pass all connectathon tests (except posix byte range locking) by this summer (by next test event at Microsoft)
  - Rename and delete of open files, acl support, hardlinks, symlinks need to be updated
- SMB2.1 will be the default dialect (current plan is only focus on 2.1 and SMB 3, may skip SMB2) which will greatly improve performance to Windows 7, Windows 2008R2, Samba
- Fewer mount options than cifs (simpler), and more strict defaults
- Priorities (in rough order):
  - Update workflow to make merges earlier in process, visible to linux-next faster
  - Highest priority code cleanup changes:
    - e.g. Remove all “if server->is_smb2” checks (replace with function pointers)
  - Pass all connectathon tests (except posix locking) with SMB2.1
  - Investigate xfstests over SMB2.1 and CIFS mounts for automated build verification
  - Cached File Performance optimizations (lease upgrade/downgrade)
  - Server side file copy (“copychunk”) and “copy-on-write” files (with user space tools)
  - Document/implement SMB2 Unix Extensions (pass POSIX byte range locking tests)
  - Pass all connectathon tests with SMB2.2 negotiated
  - Use SMB 3 Directory oplocks for better metadata caching
  - Cluster Failover
  - SMB 3 multichannel
  - SMB 3 over RDMA (will need hardware, priority could go up if have hardware)

Not For Windows Only: “SMB2 Unix Extensions”

- Allow perfect POSIX semantics when mounted to Linux
- SMB2 provides almost all features needed for POSIX compatibility, but we need minor extensions to get exact file semantics from Linux, Solaris, MacOS and other POSIX clients (in particular “advisory” POSIX locking)
  - NB: Real world Linux compatibility requires more than posix (e.g. SELinux xattrs and POSIX ACLs could be useful in some types of Linux systems)
- 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 could be implementable by Windows and others non-Unix/Linux servers
- Work in progress (with MS) to SMB2 Unix Extensions (fortunately can be smaller in scope for SMB2 than what was done for CIFS POSIX Extensions). Chris Hertel (RedHat) is helping coordinate this
- See http://www.unixsmb2.org/docs.php for information about these extensions

Historic shift occuring – What do we in Linux have to do?

- While we implement SMB 3 et al: Focus on our strengths (e.g. clustering and RDMA)
  - Samba has had GREAT clustering support for years (Tridge's ctdb was brilliant!)
  - Our RDMA stack is good, with broad driver support
  - Our server side copy (copychunk and T10 block copy) should be able to excel
  - We can be faster than Windows and non-Linux NAS
- Ensure SMB 3 Unix Extensions include all key Linux needs (POSIX locking, SELinux …)
  - Intend Unix Extensions to be small. Will be documented (see Chris Hertel at RedHat)
- Optimize BOTH ends of our Linux workloads (and not just for Apache, KVM)
  - Ensure I/O sizes, flow control are optimized on both SMB2.2 kernel client and Samba
  - Fix some VFS and client i/o bottlenecks so KVM can scale better on SMB2.2 mounts
- As Microsoft has shown with Hyper-V in Windows 8 – fixing migration of running images (KVM in our case) over SMB 3 is very powerful feature (may require VFS changes)
- Merge RichACLs so NFSv4 (4.1 actually) and Samba can get consistent access control
- Continue to take advantage of GREAT detailed protocol documentation on various loosely related protocols, and implement services, extensions and libraries to interoperate with branch cache (v2), Witness protocol (HA), Claims Based Access control (Hits Kerberos padata field, and RichACLs), and some of the CIM-like storage management features

Thank you for your time!!

Backing Charts – Older Material which may be of interest to some

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

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
- Will be even more useful if we can tie this with the “branche cache” protocol
- May also be useful for disconnected operation, or on networks which crash a lot

History of SMB2 Kernel Client

- Prototype begun about 3 years ago then rewritten a year later
- Found various spec problems and bugs at MS test events
- Work restarted 1/2010 with Jeremy Bongio helping (improved read support)
- Pavel in Google Summer of Code last two summers - fast async write and leases".
- Tested at 2009, 2010, 2011 Microsoft plugfests, and SNIA SDC 2010, 2011
- Tested at 2010 and 2011 Connectathon (many file operations working, added hardlinks and rename and jra fixed some server bugs in those areas too)
- Cthon 2011 decision to merge with common code into cifs.ko (from distinct smb2.ko) big rewrite needed.

New Design

- SMB2 prototype was in distinct git tree (smb2.git on kernel.org) and built distinct kernel binary smb2.ko (and created pseudofiles in /proc/fs/smb2)
- At SDC 2010 a “cifs_common” (kernel library common to smb2 and cifs) prototyped - reduced smb2 code size about 5% (e.g. connection establishment) was expected to grow to 20%+ common code
- To address comments from Jeff Layton and others at Connectathon 2011 change from two binaries (smb2.ko and cifs.ko) to one (cifs.ko)
  - Smb2 (mostly) in distinct C files in fs/cifs – only built when experimental SMB2 support explicitly requested in build. “vers=smb2” on mount not new fstype (-t smb2)
  - Good: increases common code dramatically. Makes it easier to review for those familiar with cifs. Longer term maintenance easier
  - Bad: requires lots of rework, many trivial changes, much higher risk of smb2 changes breaking cifs, cifs global structures grow slightly to accommodate greater smb2 limits

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
- Set better default mount options than cifs