Christian Ambach
IBM Research & Development, Samba Team

Integrating Samba and GPFS - status and outlook

Why use GPFS for CIFS serving?

- It is “General Parallel File System”, not “General Purpose File System”
- It origins are in the HPC area
  - used on lots of the supercomputers on this world
- It is very mature
  - development started in 1993, nearly 20 years ago
- Best tested file system for CTDB clusters
- It offers great scalability
  - 2^99 bytes file system size
  - 2^64 files per file system
  - 256 file systems per cluster
- It gives you great performance
  - At least for HPC-like workloads (large files, streaming IO)
- Large set of features
  - Snapshots
  - File sets (logical partitioning of file system)
  - POSIX / NFSv4 ACL support
  - ...

How to setup a CIFS server with Samba and GPFS

- Take a stock Samba build
- Put all CIFS specific information into extended attributes
  - Windows attributes
    - Use GPFS fast EA (available since GPFS 3.4)
  - use `acl_xattr` module if you want full fidelity ACLs for the clients
    - switch filesystem to POSIX ACL mode to avoid problems with NFSv4 ACLs
- Add `shadow_copy2` module if you want to present GPFS snapshots to your Windows clients
- To improve performance, turn off the protocol interoperability features like `posix locking` and `kernel oplocks` in the Samba configuration

YOU ARE DONE!

Why is there a need for Samba and GPFS integration?

There are some compelling reasons that drive the integration efforts

- Enable the use of non-standard POSIX features of GPFS
  - WAN cache ("Advanced File Management")
  - HSM (files migrated off to tape)
- Cross-protocol interaction: share the same data out via CIFS and NFS
  - CIFS share modes / NFSv4 reservations
  - CIFS oplocks / NFSv4 delegations
  - CIFS ACLs / NFSv4 ACLs
  - Data integrity must be ensured
- Better performance

The beginning (2006)

GPFS 3.1

- Initial release containing libgpfs_gpl.{h,so}
- Library offered interfaces for
  - ACLs
  - Sharemodes (READ, WRITE, no DELETE)
  - leases (oplocks/delegations)
  - pre-allocation

Samba

- vfs_gpfs.c was born late in 2006
- Contained global options to control which GPFS calls to use or not
- started with oplock and sharemode support
- Next came ACL support

Moving on with the initial feature set (2007)

GPFS 3.2

- libgpfs.so completely available under BSD-style license
- gave Samba access to all other functions in the library

Samba

- ACL work
- more ACL work
- even more ACL work

Proceeding with slow pace (2008)

GPFS

- no new major release
- 3.2 PTF brought new functions:
- `gpfs_get_realfilename_path()`
- `gpfs_ftruncate()`

Samba

- Adopted to new libgpfs.so license
- Small ACL improvements
- Exploited `gpfs_get_realfilename_path()`
- and `gpfs_ftruncate()`

2009 – GPFS 3.3

GPFS 3.3

- Windows attributes
- Object birthtime support
  – stored in the Windows attributes
- added missing SHARE_DELETE

Samba

- Exploited Windows attributes
- Exploited birthtime
- ACL fixes

2010 – GPFS 3.4

GPFS 3.4

- Added ACE_FLAG_INHERITED
- “fast” Extended Attributes
- added `gpfs_lib_init()`
- PTF brought Bypass Traverse Check in 2011

Samba

- only small fixes
- No explicit exploitation of gpfs_lib_init() or “fast” EA required

2011 (no new GPFS release)

Samba

- Exploit NFSv4.1 INHERITED_ACE ACL bit
  - Gives much better reporting of inheritance in Windows Explorer
  - Fixes up some ACL issues with Explorer
- More ACL fixes
- converted most module options to be per-share options
- Use `gpfs_lib_init()`
- Added `gpfs:syncio` option
  - Open all files in a share with O_SYNC
  - helps with performance for certain workloads
  - Increases data integrity in error cases
- Interpret offline bit in GPFS Windows attributes (`gpfs:hsm`, `gpfs:winattr`)
  - For WAN cache and HSM
  - Enforce AIO for offline files
  - No need for `vfs_tsmsm` on GPFS any more
  - Needed to add `gpfs_hsm_notify` to send notifications to because of non-stackable AIO paths

The last few months (2012)

Samba

- Use GPFS quota for free-space reports (`gpfs:dfreequota`)
  - Gives users better indication of how much space they have left on a share
  - Works with user/group and fileset quota
  - Also good for legacy applications that cannot deal with 50 TB of free space
- Preallocation
  - Exploit `gpfs_prealloc()`
    - Without it, performance with `strict allocate = yes` will be horrible due to glibc emulation
    - depended on the fallocate improvements that Jeremy added in late 2010
    - `strict allocate` can be set to yes now
  - Fixes performance problems with some applications like MS Access

Summary of the current state

GPFS version	Interface	Exploited in Samba?
3.1	ACLs	yes
3.1	Sharemodes (READ, WRITE)	yes
3.1	Leases (oplocks/delegations)	yes
3.1	pre-allocation	yes
3.2	gpfs_get_realfilename_path()	yes
3.2	gpfs_ftruncate()	yes
3.3	Windows attributes	yes
3.3	Birthtime	yes
3.3	SHARE_DELETE	no
3.4	ACE_FLAG_INHERITED	yes
3.4	gpfs_lib_init()	yes

Current limitations

- No Level II oplocks
  - Due to Linux kernel limitations, downgrading to Level II oplocks does not work and so Samba does not give out Level II oplocks
  - GPFS lease code is based on Kernel implementation, so same restrictions apply
- Open / share-mode / lease races
  - Need to be resolved for NFSv4 serving
  - Currently only Samba sets sharemodes and requests leases, so no races for that can happen
- Enabling SMB2 requires GPFS 3.4 or 3.5 due to `rename()` problem
  - or use `gpfs:sharemodes = no` for older GPFS releases

GPFS share-modes and Samba

GPFS cannot read Samba's mind – GPFS developer

- When a share-mode has been set on a file, GPFS will act as told and refuse any conflicting request (even for the root user)
- Problems arise with system calls like `unlink(char *path)` or `rename(char *old, char *new)` that operate on paths, not open instances
- GPFS cannot determine to which open file descriptor such a call refers to
- Some GPFS internal changes required also special handling of `ftruncate()` since GPFS 3.2.1.11, although that is a file descriptor based call (but touches the inode information)
- So special code was added to `vfs_gpfs.c` over time
  - before calling `unlink()`, all share-modes are dropped (racy!)
  - for `ftruncate()`, use special `gpfs_ftruncate()` call
  - problems showed up with SMB2
    - file rename in SMB2 works on open instance
    - rename() was blocked by GPFS
      => could not save a file with Microsoft Word anymore!
    - GPFS added configuration option in 3.4 PTF to skip sharemodes checks on rename for Samba

ACL gotchas

- If your altitude is high enough, NFSv4 ACLs look similar to Windows security descriptors, e.g. ACE bits are similary named
- But semantics are different!
  - Example: NFSv4 READ_ATTR: *The ability to read basic attributes (non-ACLs) of a file.On a UNIX system, basic attributes can be thought of as the stat-level attributes.* vs. FILE_READ_ATTRIBUTES: The ability to read the Windows attributes, stat-level information is always available on Windows
- So we are losing details here that might be important!
- Why not use the acl_xattr module to map to NFSv4 as good as we can and still store complete SDs and evaluate them for CIFS clients?
  - Sometimes the current NFSv4 mapping generates ACLs that are more restrictive than they should be
  - So if acl_xattr says that access would be OK, GPFS might still block it

New pieces in GPFS 3.5

- Adds sideband channel for Samba
  - Will allow for atomic CreateFile() equivalent
    - Ask for sharemode / oplock and initial ACL during open()
  - Will fix the share mode issues with system calls that do not take a file descriptor

-
  ```
  typedef struct cifsThreadData_t
     {
       unsigned int dataLength; /* Total buffer length */
       unsigned int share;      /* gpfs_set_share declaration */
       unsigned int deny;       /* gpfs_set_share specification */
       unsigned int lease;      /* gpfs_set_lease lease type */
       unsigned int secInfoFlags; /* Future use.  Must be zero */
       gpfs_uid_t   sdUID;      /* Owning user */
       gpfs_uid_t   sdGID;      /* Owning group */
       unsigned int aclLength ; /* Length of the following ACL */
       gpfs_acl_t   acl;        /* The initial ACL for create/mkdir */
     } cifsThreadData_t;
  ```

- Samba needs to fill out the struct and register it with GPFS before doing the systemcall like open()

New pieces in GPFS 3.5 (Part II)

- New ACL bits
  - allows the storing of ACL specific bits like DACL_PROTECTED

-
  ```
  #define ACL4_FLAG_OWNER_DEFAULTED               0x00000100
     #define ACL4_FLAG_GROUP_DEFAULTED               0x00000200
     #define ACL4_FLAG_DACL_PRESENT                  0x00000400
     [...]
  ```
- Makes it possible to store all bits that are in a Windows SD into a GPFS ACL now
- This is the first step, future enhancements in GPFS will also make GPFS actually understand and act on the bits
- Still does not allow to store ACEs that refer to a non-mappable SID (e.g. workstation SIDs)

New pieces in GPFS 3.5 (Part III)

- Independent file-sets
  - They can have their own snapshots
  - Use `shadow:snapdirseverywhere = true` option of shadow_copy2 module
- “stat() lite”
  - Makes it possible to request stat() information with given accuracy (size, timestamps)
  - In addition to normal `struct stat`, it will return birthtime and Windows Attributes

Things that need to be cleaned up

- Windows attributes / birthtime
  - Birthtime is stored in the GPFS windows attributes structure
  - Current integration is a kludge: vfs_gpfs intercepts xattr calls that are used by `store dos attributes` and en-/decodes the EA blob and does conversion to the GPFS library calls
- Use `gpfs_get_realfilename()` properly
  - Currently, Samba calls GPFS in the wrong way and GPFS always gives negative answer
  - So benefits of the call do not apply, Samba still might have to guess the case of the file
- Simplify some code paths that act differently dependent on GPFS version
  - Remove support for GPFS 3.1 and GPFS 3.2 as they are out of IBM's support

ToDo List

- Further improve ACL compatibility
  - Store CIFS ACLs as they are in GPFS?
  - Sort out semantical differences
- Fix up AIO in Samba to allow stacking
  - removes the need for the gpfs_hsm_notify module
- SMB2.x / SMB 3.0 support
  - (directory) leases
  - Durable / Persistent handles
- Use stat() lite
- Cross-protocol file change notify
- Alternate data streams?

Q & A

Thanks!