Redefining CephFS bridge with the new VFS module for Ceph

`vfs_ceph_new`

Anoop C S

IBM / Samba Team

April 8, 2025

SAMBA EXPERIENCE

Agenda

Agenda

Samba and VFS

- Free software *re*-implementation of **SMB** networking protocol
- File and print services and more...
- Integration of AD (Active Directory)
  - DC (Domain Controller)
  - Domain member
- Stackable **VFS** interface
  - Local - btrfs, ext4, xfs etc.
  - Clustered - cephfs, glusterfs, gpfs etc.
- Modern **VFS**
  - File handles
  - Path reference `fsp`
  - `O_PATH` in Linux

Ceph and CephFS

- Unified system for object, block and **file** storage
- Highly reliable, easy to manage and free
- CephFS
  - **POSIX** complaint file system
  - Built on top of Ceph's distributed object store **RADOS**
  - **MDS** (MetaData Server)
  - Data and Metadata pools
- Convenient client facing libraries

Agenda

Userspace CephFS API

- `libcephfs` - native C API
- Easy to use
- Correspondence to standard file systems calls
  - `ceph_openat()` = `openat(2)`
  - `ceph_readdir()` = `readdir(2)`
- Included as `libcephfs.h`
  - Versioned
  - Provides *pkgconfig* file: `cephfs.pc`
- API behaviour changes not properly versioned

`libcephfs` versioning

Currently versioned at **2.0.0**

High and Low

Two variants for LibCephFS

High level API

- Most commonly used
- Matches standard function signatures
- **Path** and **File descriptor** based
- Support for `*at()` calls

Examples

`ceph_openat()`, `ceph_close()`, `ceph_getxattr()`

Low level API

- More fine grained
- Notion of **inodes** and **file handles**
- Support for user permissions (per call) and supplementary groups
- **Asynchronous IO**

Examples

`ceph_ll_lookup()`, `ceph_ll_open()`, `ceph_ll_put()`

Agenda

Previously..

- Existing vfs module for ceph: `vfs_ceph`
  - Long standing and it works !
- Source: `source3/modules/vfs_ceph.c`
- Consumes **high level** `libcephfs` APIs
- Implements most of the `SMB_VFS_XXXX` interfaces
  - Including path based fallback mechanism
- Statically linked against `libcephfs.so`
- Mandatory to be loaded as the **last module** in `vfs objects` list
- File descriptors maintained on `libcephfs` client side
  - fds are passed over as it is to upper layers

The trigger

- BUG 14053 *vfs_ceph (libcephfs) denies access when permitted via supplementary group membership*
  - Shachar Sharon created MR 3466
  - Proposed changes addressed the direct issue
  - But the ability for `smbd` to switch credentials was implicitly ignored
- Further discussions and looking for alternatives

Thoughts outlined

1. Shift to **low level** `libcephfs` APIs
2. Additionally **optimize** CPU and memory consumption due to separate independent `libcephfs` client stacks

A name?

- Should we make changes to `vfs_ceph`?
  - not minimally invasive
- Separate module? perfect.
- What about a name? uh-oh !
  - `vfs_ceph_ll`
  - `vfs_ceph2`
  - `vfs_ceph_experimental`
  - `vfs_ceph_test`
  - `vfs_ceph_next`
  - `vfs_ceph_ng`
- Finally settled on **vfs_ceph_new**

vfs_ceph_new structure

- Source: `source3/modules/vfs_ceph_new.c`
- Initially crafted out of `vfs_ceph` MR 3718
- CephFS client initialization with locally **cached mounts**
  - Helpful for access to multiple shares configured with same ceph user and underlying ceph file system from a unique client.
- Operations on **inodes** via **file handles**
  - Leverages vfs fsp extensions to hold `struct vfs_ceph_fh`
  - Proactively adds credentials using `get_current_utok()`
- No longer statically linked against `libcephfs.so`

struct vfs_ceph_fh

```
/* Ceph file-handles via fsp-extension */
struct vfs_ceph_fh {
    struct vfs_ceph_dirp dirp;
    struct cephmount_cached *cme;
    struct UserPerm *uperm;
    struct files_struct *fsp;
    struct vfs_ceph_config *config;
    struct vfs_ceph_iref iref;
    struct Fh *fh;
    struct dirent *de;
    int fd;
    int o_flags;
};
```

Expanding further

- Introduction of **proxy** mode MR 3792
  - Structural reorganization, new `struct vfs_ceph_config`
  - Abstraction of all module specifications (both internal and external)
  - Dynamic loading of libraries (`libcephfs.so` or `libcephfs_proxy.so`)
- Switch to use `ceph_readdir_r()` from `ceph_readdir()` MR 3833
- Making use of low level **non blocking** API for asynchronous IO MR 3857
  - Detection of API during compile/build time
  - `struct tevent_threaded_context` added
- Taking advantage of **case insensitive** CephFS subvolumes MR 3992
  - Identify and conditionally add `FILE_CASE_SENSITIVE_SEARCH` flag

struct vfs_ceph_config

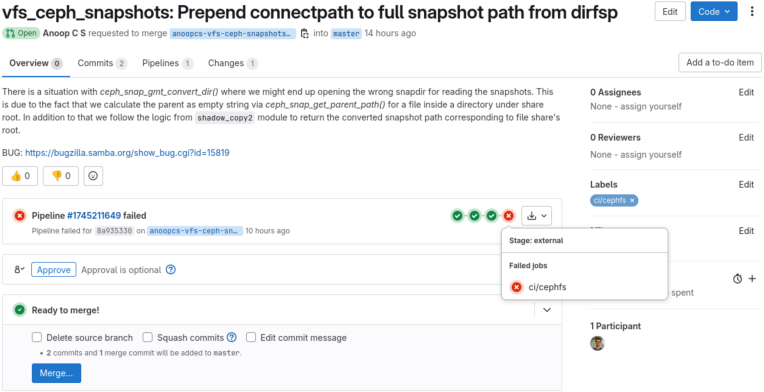
```
struct vfs_ceph_config {
#if HAVE_CEPH_ASYNCIO
    struct tevent_threaded_context *tctx;
#endif
    const char *conf_file;
    const char *user_id;
    const char *fsname;
    struct cephmount_cached *mount_entry;
    struct ceph_mount_info *mount;
    enum vfs_cephfs_proxy_mode proxy;
    void *libhandle;
    uint32_t capabilities;
    // ...
};
```

Agenda

CI/CD pipeline

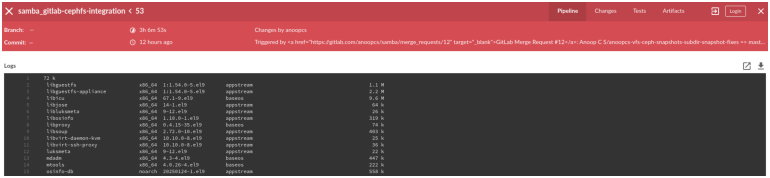
- Modular ansible based testing framework
  - Samba Integration Testing (SIT)
  - Refer sambaXP 2024 talk
- Possible integration for Samba GitLab CI?
- Includes a subset of core *smb2* torture test suites
- Various share combinations for CephFS backend
  - ceph kernel client based shares
  - `vfs_ceph`
  - `vfs_ceph_new`
  - `vfs_ceph_new` under proxy mode

GitLab integration (projected view)

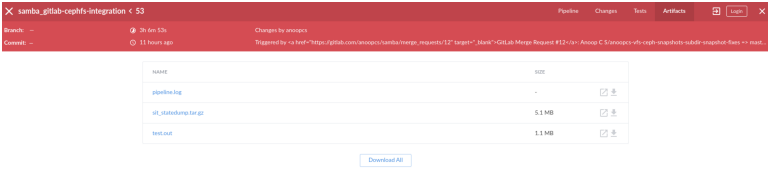


Sample MR

Jenkins redirected (projected view)



Jenkins job landing page



Jenkins job artifacts page

Agenda

Findings

Functionality wise

- Test failures especially `smbtorture` failures
- Interesting ones
  1. `smb2.charset` test suite failure BUG 15716
     - Caused by *re*-organization around retrieving FS capabilities
  2. Ceph MDS crash tracker 69059
     - Somehow triggered a different code path by `smbd` cleanups/changes
  3. LibCephFS crash tracker 69624
     - Related to snapshot lookups when stacked with `vfs_ceph_snapshots`
  4. `smb2.rw.rw1` torture test failure tracker 70726
     - Incorrect behaviour during asynchronous IO, still open

Note

`smbtorture` was found to be a good bug detector !

Case (in)sensitive CephFS subvolumes

What is the difference?
Server versions: Samba 4.21.4 + Ceph 19.2.1

Case sensitive	Latency	Throughput	Ops rate
Enabled	12.637 ms	2189.222 KB/s	316.540 ops/s
Disabled	12.261 ms	2256.935 KB/s	326.219 ops/s

Workload: SWBUILD

Note

- Benchmark utility: SPECstorage
- Case sensitivity set at subvolume level using `ceph fs subvolume charmap set`

Asynchronous IO

What about performance?
Server versions: Samba 4.21.4 + Ceph 19.2.1

IO mode	Latency	Throughput	Ops rate
Sync IO	12.261 ms	2256.935 KB/s	326.219 ops/s
Async IO	11.877 ms	2332.388 KB/s	336.736 ops/s

Workload: SWBUILD

Note

- Benchmark utility: SPECstorage
- Backed by case **in**sensitive CephFS subvolume
- `aio read size` and `aio write size` parameters from *smb.conf* controls IO mode

Agenda

Roadmap

- Optimized caching of ceph mounts
  - Integrate `struct cephmount_cached` with **memcache**?
- **Zero-copy** interface
  - Under active development PR 62003
- Profile memory usage, probably detect leaks
- Optional **extra pipeline** job for upstream GitLab MRs
  - Run tests on multiple share configurations backed by CephFS
  - Propose to `samba-technical`

Agenda

Thank you

Anoop C S

anoopcs@samba.org

Questions?