Bridging Object and File: Implementing SMB Access to Ceph RGW via vfs_ceph_rgw

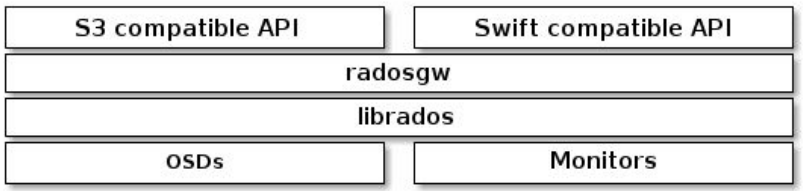
Vinit Agnihotri
(IBM India)

Agenda

- Ceph object gateway
- Need for smb over rgw
- Using librgw
- Module details
- Implementation challenges
- Current status
- Future scope and limitations
- Demo

Ceph object gateway

- Object storage interface built on top of librados
- Support 2 interfaces s3-compatible and swift compatible
- Ceph Object Gateway daemon (radosgw)
- Advantages:
  - Scalability and Perormance
  - Industry standard APIs
  - Large scale deployments



Why SMB over RGW

- Legacy application support
- Ease of use for clients
- Bridging protocols
- File level management
- Familiar access control

Solution?

- Shared library: librgw
  - Ceph Object gateway instance
- Namespace conventions
  - Confomrs to AWS hierarchical namespace conventions.
- Security
  - RGW/S3 security credentials
- Supported operations
  - Standard posix operations
- Limitations
  - Advance ACLs
  - Directory renaming
  - Random writes

vfs_ceph_rgw

- Dependencies
  - Librgw2-devel
- Samba build config option
  - —enable-cephrgw
- Module parameters
  - path = /
  - aio write size = 0
  - vfs objects = ceph_rgw
  - ceph_rgw: bucket = <name of bucket>
  - ceph_rgw: user_id = <object store user id>
  - ceph_rgw: access_key = <access_key for user_id>
  - ceph_rgw: secret_access_key = <secret_access_key for user_id>
  - ceph_rgw: config_file = <ceph_config file path> (default: /etc/ceph/ceph.conf)
  - ceph_rgw: keyring_file = <ceph_keyring_path> (default: /etc/ceph/ceph.client.admin.keyring)
  - ceph_rgw: debug = off/on
- Forced params
  - vfs mkdir use tmp name

Challenges

- Paths relative to bucket root
  - Handling of special names (. and /)
- Handle based operations
  - int rgw_lookup(struct rgw_fs *rgw_fs, struct rgw_file_handle *parent_fh, const char *path, struct rgw_file_handle **fh, struct stat *st, uint32_t mask, uint32_t flags);
- File creation
  - int rgw_create(struct rgw_fs *rgw_fs, struct rgw_file_handle *parent_fh, const char *name, struct stat *st, uint32_t mask, struct rgw_file_handle **fh, uint32_t posix_flags, uint32_t flags);
- Directory creation
  - rgw_create() / rgw_mkdir()
- File open/Directory open
  - int rgw_open(struct rgw_fs *rgw_fs, struct rgw_file_handle *parent_fh, uint32_t posix_flags, uint32_t flags);
- Directory listing
  - int rgw_readdir2(struct rgw_fs *rgw_fs, struct rgw_file_handle *parent_fh, const char ***token**, rgw_readdir_cb **rcb**, void *cb_arg, bool *eof, uint32_t flags);
    - rcb (const char *name, void *arg, uint64_t offset, struct stat *st, uint32_t mask,uint32_t flags);
    - token

Current status

- Basic operations
  - Openat, close, pread, pwrite, rename, unlink, stat, lstat, fstat
- Directory operations
  - Open, close, mkdir, readdir, rewind
- Other operations:
  - get/set xattrs, fntimes, get/set dos attrs
- Testing
  - Server
    - Centos9 stream vm running ‘ceph tentacle’ cluster on single node
  - Client
    - fedora42 packaged smbclient, Windows 10 workstation
  - Read/Write/List
    - r/w/verify files with various sizes, create and list 1000+ files/folders
- Upstream MR
  - https://gitlab.com/samba-team/samba/-/merge_requests/4381
  - Authors
    - Vinit Agnihotri, Shachar Sheron

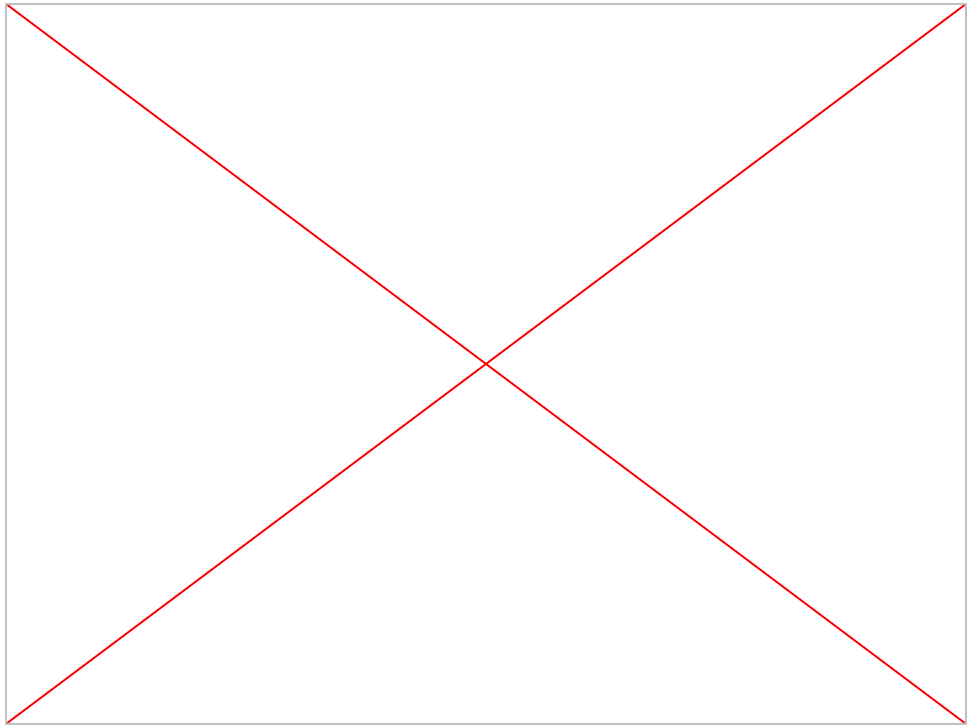
Future scope and limitations

- Future scope
  - Make module at par with NFS RGW FSAL module.
  - Proxy module for scalability
- Limitations
  - Random writes
  - Directory rename
  - Change notifications
- Known issues
  - Renaming file/directory triggering unlink in backend
  - Inconsistent callback for getxattr

Demo time

- Create objects using aws APIs
- Read objects with smbclient
- Write files with smbclient
- List directory having 1000+ entries
- Create nested structure for files and folders
- Browse share

Demo time



Thank you.

Questions?