Scaling Ceph-SMB Connections

Sachin Prabhu
Avan Thakkar
IBM/Red Hat

Introductions

- The team
    - IBM / Red Hat
    - Ceph team - SMB service
        - https://github.com/samba-in-kubernetes

Introductions

- Ceph-SMB service
  - smb manager module
  - container - samba-container project
  - exports cephfs volumes
  - samba vfs module - vfs_ceph_new

Samba

- The Forking model
  - portability
  - switch uid/gid of running process
  - robustness

Problem

- Large number of simultaneous clients
  - large number of processes
  - each connection has its own libcephfs stack
    - own metadata and data cache
  - leads to depletion of resources for some workloads

Reproducer

- sit-test-cases - loading test
  - https://github.com/samba-in-kubernetes/sit-test-cases
  - smbprotocol python module
  - multiple threads each opening a new client connection
  - multiple files opened/closed
  - 16 M file size
- fails after 100 simultaneous connections
  - failure caused by memory pressure

Solution

- libcephfs_proxy
- design document in ceph repo
  - doc/dev/libcephfs_proxy.rst
- avoid an independant cache for each client connection
- tested with 1000+ simultaneous connections
- 2 parts
  - libcephfsd daemon process
  - libcephfs_proxy.so library

Solution

- libcephfsd daemon
  - uses actual libcephfs.so library to connect to cephfs volume
  - centralise libcephfs requests
  - listens to incoming connections from the client at unix socket
    - /run/libcephfsd.sock

Solution

- libcephfs_proxy.so library
  - provides a subset of low level cephfs API calls
  - to be used in place of libcephfs.so
  - no caching on client
  - forwards requests to libcephfsd daemon over unix socket

Solution

- Same configurations share connection
- Some calls need special handling - getcwd, chdir

Solution

```
[share]
path = /volumes/_nogroup/shares/bbd11c17-ae54-4d98-9a99-5…
vfs objects = acl_xattr ceph_new
ceph_new: config_file = /etc/ceph/ceph.conf
ceph_new: user_id = samba_dev
..
ceph_new:proxy = yes
```

Solution

- libcephfs_proxy.so, libcephfsd to be installed
- Modify smb.conf to enable proxy
- Start libcephfsd daemon
  - listens on /run/libcephfsd.sock
- Start smbd

Performance implications

- SPECstorage - Performance tests
  - CTDB enabled
  - cifs kernel mount
  - Ceph 19.2.0-10, Samba 4.21.0
- Higher Latency
  - SWBuild 89.708 ms vs 140.095 ms
  - VDA 75.933ms vs 97.330 ms
- Overall throughput decreased
  - SWBuild 1438.143 kb/s vs 917.124 kb/s
  - VDA 23001.164 kb/s vs 22817.778 kb/s

Future Plans

- Metadata cache on client end
  - requires synchronous invalidation callbacks from ceph
- Consider other options for connection between libcephfs_proxy.so and daemon process
- Extend low level API calls supported
- Handling libcephfsd crashes. Reconnections should be transparent to the clients.

Improving Performance

- async io
  - vfs_ceph_new
  - libcephfs proxy
  - negotiation & async callbacks for communication between proxy & daemon
- case sensitivity

Performance improvements

- Latency
  - SWBuild 89.708 ms vs 140.095 ms vs 119.959 ms
  - VDA 75.933ms vs 97.330 ms vs 75.548 ms
- Throughput
  - SWBuild 1438.143 kb/s vs 917.124 kb/s vs 1072.184 kb/s
  - VDA 23001.164 kb/s vs 22817.778 kb/s vs 23065.546 kb/s

Metadata Caching

- Metadata cache on client end
  - Why metadata caching is needed in libcephfs_proxy
  - How it can be implemented
  - Challenges
  - Possible solutions & improvements

Why Metadata Caching

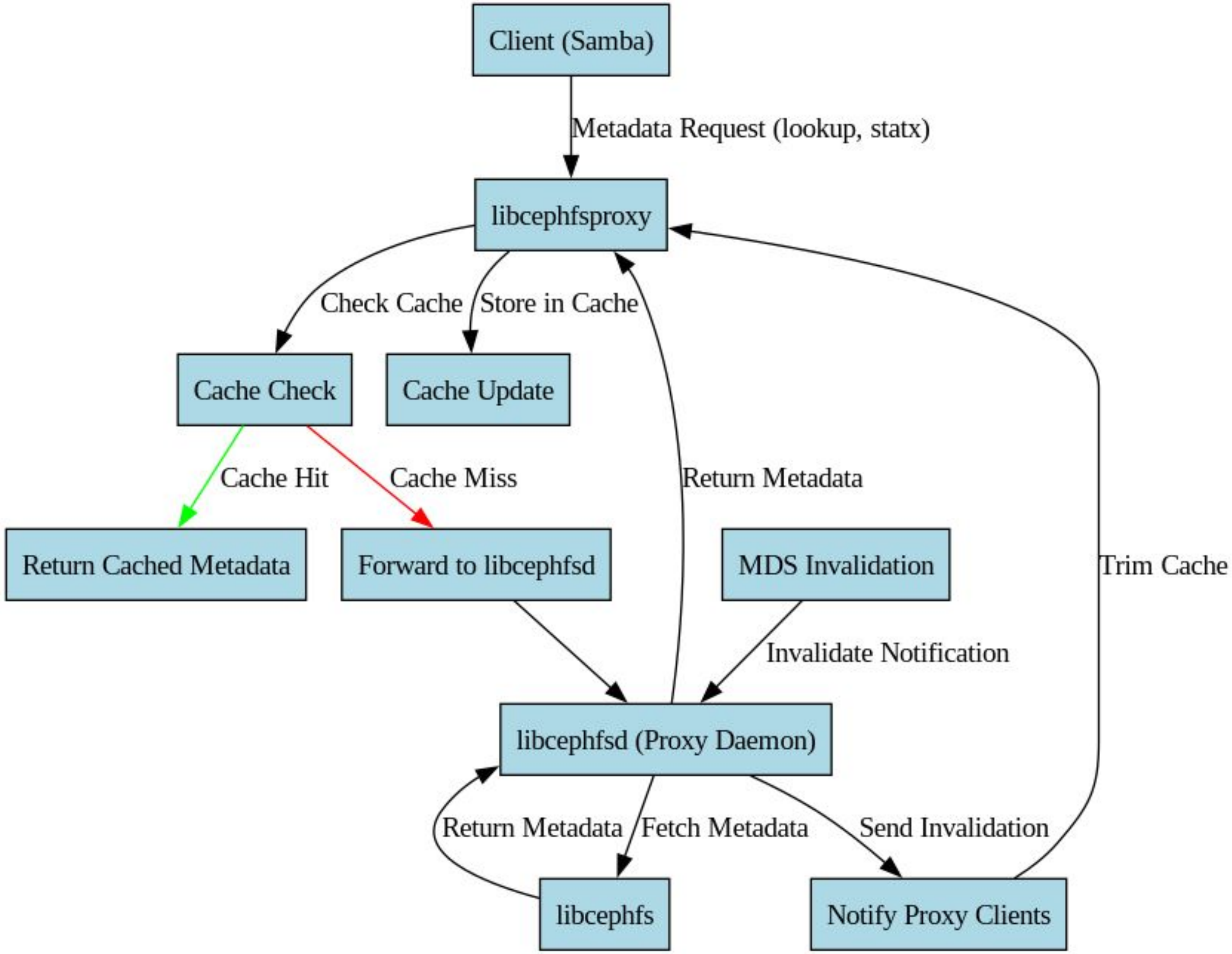
🎯 Problem: High Overhead for Metadata Operations

- Each metadata operations (**statx, lookup, getattr**) **requires an extra hop** (Proxy → Daemon → libcephfs → MDS).
- **Fewer metadata requests** reaching the proxy reduce its load, allowing it to respond faster to other requests.

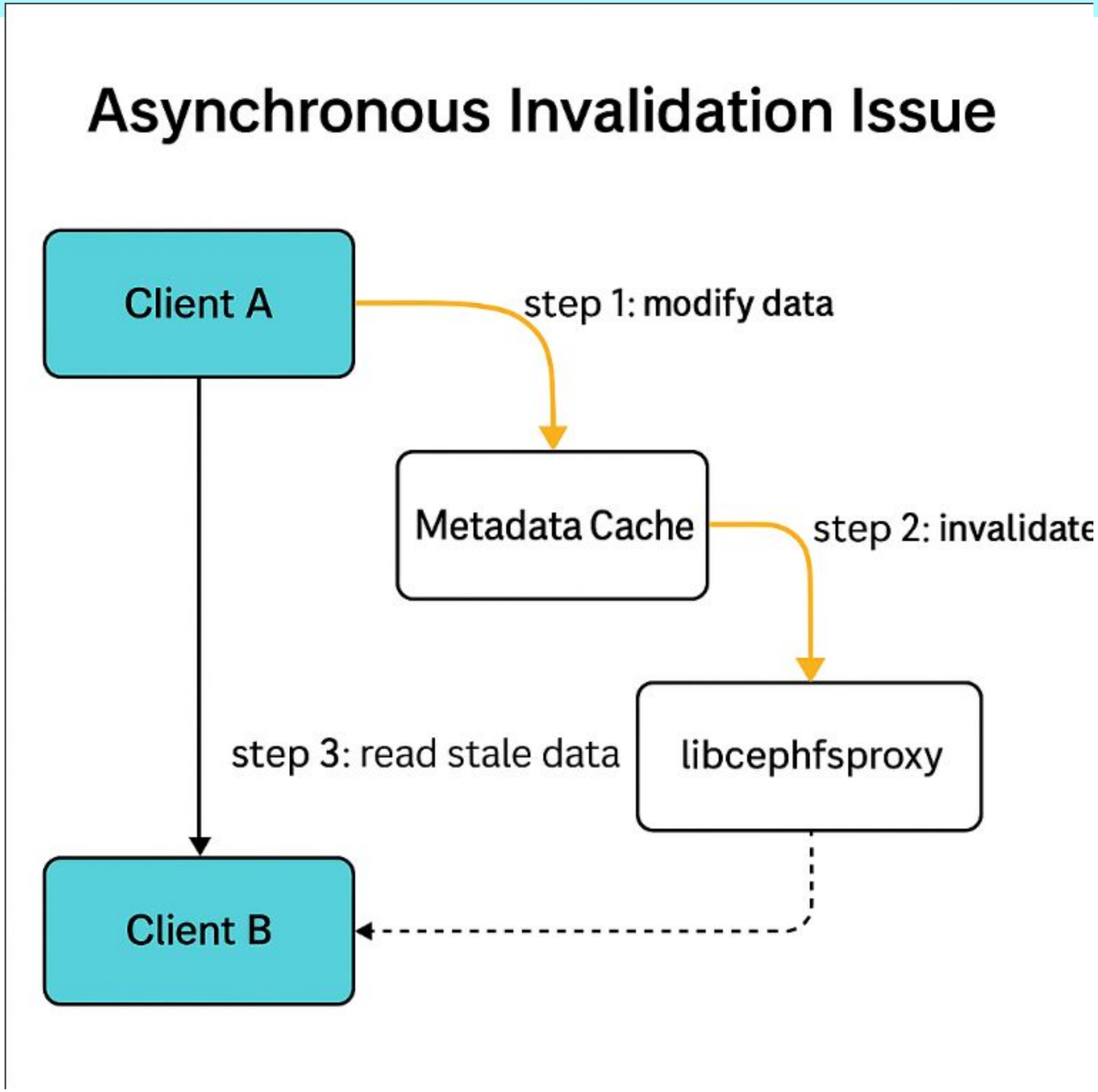
💡 Solution: Introduce metadata caching in `libcephfs_proxy`

- Cache metadata for **frequent statx & lookup calls**.
- Avoid unnecessary trips to the daemon when metadata is unchanged.

How it works


Client (Samba)
Metadata Request (lookup, statx)
libcephfsproxy
Check Cache
Store in Cache
Cache Check
Cache Update
Cache Hit
Cache Miss
Return Metadata
Return Cached Metadata
Forward to libcephfsd
MDS Invalidation
Trim Cache
Invalidate Notification
libcephfsd (Proxy Daemon)
Return Metadata
Fetch Metadata
Send Invalidation
libcephfs
Notify Proxy Clients

Challenges with current approach



Possible Solution

📌 **Fully Synchronous Invalidation**

- Ensures that an invalidation is acknowledged by all proxy clients before the original modifying request completes
- **Problem**: The original request must wait for all proxy clients with cached metadata to acknowledge invalidation. With thousands of clients, this delay can become significant.

📌 **Two-Phase Invalidation (Proposed by @Xavi Hernandez)**

- **Phase 1 (Asynchronous)**: The proxy-daemon starts an invalidation when **caps are dropped** by the client
- **Phase 2 (Synchronous)**: The client calls a final **synchronous** invalidation callback once the MDS provides updated metadata
- This approach allows proxy clients to start processing invalidation *before* the final invalidation request arrives. By the time the synchronous invalidation occurs, many clients may have already invalidated their cache, reducing the overall latency of completing the process.

🎯 **Actively in talks with the CephFS team** to improve invalidation callbacks.
Upstream Ceph tracker: https://tracker.ceph.com/issues/69761

Thank you

Sachin Prabhu - sprabhu@redhat.com

Avan Thakkar - athakkar@redhat.com