Speeding Up CephFS Mirroring with Bounded-Frontier Concurrency & Parallel I/O

How CephFS Mirroring Works (High Level)

- Two clusters: Local (source) → Remote (target)
- Register directories with "ceph fs snapshot mirror add <directory>"
- Goal per directory: replicate its .snap timeline from Local to Remote
- Local snapshots (example): $snap_1$, $snap_2$, $snap_3$, $snap_4$
- Idea: advance Remote by applying deltas between adjacent snapshots

Delta Application (Concept)

Define $\Delta_i = snap_{i+1}(local) - snap_i(local)$

- **Apply in order:**
    - $Remote@snap_1 = Remote@base + (snap_1 - base)$
    - $Remote@snap_2 = Remote@snap_1 + (snap_2 - snap_1)$
    - $Remote@snap_3 = Remote@snap_2 + (snap_3 - snap_2)$
    - $Remote@snap_4 = Remote@snap_3 + (snap_4 - snap_3)$
- **Mirror must:**
    - Read Δ_i efficiently from local cluster (adds/updates/deletes, attrs)
    - Apply Δ_i safely in remote cluster

Previous vs Current: CephFS Mirror—Problems & Fixes

Rigid concurrency (per-directory threads)

- Each registered directory spawns one thread that does everything (scan/list, delta compute, copy, deletes).
- Concurrency is fixed by directory count, not by available cores or workload.

Load imbalance & idle cores

- Uneven sizes: small directories finish early → their threads go idle while a few large dirs keep syncing.
- Mismatched snapshot schedules: when one dir snapshots first, only its thread runs; other threads sleep.

Flexible concurrency

- Concurrency will be defined by the users
- Whole syncing process in that moment will be operated by those defined threads

Load balance & no idle cores

- Defined thread's task is to evenly distribute the load of current syncing process between threads.
- Doesn't depend on how many directories registered or how many of them are currently in active synchronization. It tries to divide the whole synchronization task granularly and distribute to the threads.

Previous vs Current: CephFS Mirror—Problems & Fixes

croit

Failover/resume inefficiency

- After failover, remote may be partially synced/inconsistent.
- Current behavior often computes deltas against remote state (Δ = remote – $snap_i$ (local)) → extra remote I/O and more RTTs.
- Backup clusters typically have slower hardware/network, so this is costly and slow.

Net effect

- Underutilized CPU & bandwidth, higher tail latency, and longer recovery after interruptions.

Failover/resume efficiency

- After failover, remote may be partially synced/inconsistent.
- Only the delta regions (Δ = $snap_{i+1}$(local) – $snap_i$(local)) are re-evaluated from the remote, not the full FS tree.

Net effect

- Properly utilized CPU & bandwidth, lower tail latency, and comparatively lower recovery time after interruptions.

Previous vs Current: CephFS Mirror—Problems & Fixes

croit

Summary:

⚠️ Per-directory thread = rigid concurrency
✅ Bounded frontier + thread pools = tunable concurrency (N/M, queues)

⚠️ Load tied to directory count/size → idle threads
✅ Granular tasking spreads work evenly across threads

⚠️ Snapshot schedule mismatch → only 1 thread runs
✅ Queue admits work as ready; other threads stay busy

⚠️ Failover: delta vs remote → heavy remote I/O
✅ Snapshot-aware resume: compute delta from local; scan only deltas on remote

⚠️ Unpredictable memory with more threads
✅ Bounded queues + in-flight count caps = predictable memory/back-pressure

⚠️ Hard to tune
✅ Simple knobs: scan threads, copy threads, queue caps, in-flight bytes

Concurrency approach on high level

croit

Goal:
Break the entire mirroring workflow into small tasks and run them on thread pools that we can tune with user knobs (threads, queue caps, in-flight bytes).

Two thread pools

- **Directory metadata pool (“directory syncing”)**
  Handles metadata I/O across all registered directories:
  scan/list, stat, mkdir/rmdir, renames, deletes.
- **File data pool (“file mirroring”)**
  Handles data I/O across all registered directories:
  copy changed/new files from Local → Remote.

Why this works

- Granular tasks → balanced load (not tied to directory count/size).
- Parallel metadata + parallel data without overloading memory (bounded queues).
- User-configurable scaling: adjust scan threads, copy threads, queue capacities, and in-flight task count to match cores and bandwidth.

File Mirroring Pool — Low-Level

Role

- Copy changed/new files from Local → Remote across all registered directories.
- Runs on a thread pool with a global in-flight file count cap (F).

Task model (per file)

- FILE_TASK(directory, path, attrs)
- Steps: open/read → chunk → write → sync → verify/close
- Use aligned chunks (e.g., 1–8 MiB) to reduce syscall overhead.

Scheduling & back-pressure

- Up to M copy threads pull tasks while in_flight < F.
- When in_flight == F, producers pause → natural back-pressure, predictable memory.
- Keep one owner per socket/fd (or batch writes) to avoid lock contention.

Fairness & throughput

- Per-dir round-robin (or token bucket) to avoid hot-dir starvation.

User knobs (counts, not bytes)

- M = copy threads.
- F = max in-flight files.

Directory Syncing Pool: Scope, Tasks & Ordering - Low Level

croit

Role (metadata I/O across all registered dirs)

- List/scan (readdir), stat, mkdir/rmdir, renames, attrs
- Feeds file tasks to the File Mirroring Pool

Task granularity

- SYNC_TREE_TASK(root_dir, dir_path): syncing the sub-tree under directory dir_path
  - Create directory(if it's a new directory), synchronize stat then scan the sub-directory list from both cur_snap and prev_snap.
  - Create DELETE_TREE_TASK for directory which exist in prev_snap but not in cur_snap
  - For existing directory/newly created directory create SYNC_TREE_TASK
  - Try to push those sub-task into the queue if there is space, otherwise execute those task in the same thread space.
- DELETE_TREE_TASK(root_dir, dir_path): delete the sub-tree under directory dir_path
  - Execute recursive deletion in remote cluster

Directory Syncing Pool: Scope, Tasks & Ordering - Low Level

croit

Role (metadata I/O across all registered dirs)

- List/scan (readdir), stat, mkdir/rmdir, renames, attrs
- Feeds file tasks to the File Mirroring Pool

Task granularity

- SYNC_TREE_TASK(root_dir, dir_path): syncing the sub-tree under directory dir_path
    - Create directory(if it's a new directory), synchronize stat then scan the sub-directory list from both cur_snap and prev_snap.
    - Create DELETE_TREE_TASK for directory which exist in prev_snap but not in cur_snap
    - For existing directory/newly created directory create SYNC_TREE_TASK
    - Try to push those sub-task into the queue if there is space, otherwise execute those task in the same thread space.
- DELETE_TREE_TASK(root_dir, dir_path): delete the sub-tree under directory dir_path
    - Execute recursive deletion in remote cluster

Directory Syncing Pool: Scope, Tasks & Ordering - Low Level

Ordering & dependencies and deadlocks

- Parent-before-child rule: a child SYNC_TREE_TASKis runnable only after parent mkdir observed created or exists
- We can't just throw those task into a thread-pool regarding the memory constraints.
- If we limit the queue and use the same breadth first scanning approach it will fall into a deadlock as when queue is full it has to wait for the room in queue. But the room will only be created when current task is able to push all the sub task into the queue which falls into a classic cyclic dependency for deadlock

Bounded frontier (C) + BFS↔DFS as a solution

- Enqueue child directory SYNC_TREE_TASK when queue has headroom (BFS)
- Dive when full (DFS), opportunistically enqueue when slots free
- Keeps memory predictable; spreads work across threads

User Configurable Knobs at One Go

Concurrency

- cephfs_mirror_dir_scanning_thread — directory workers (metadata pool)
- cephfs_mirror_file_sync_thread — file-copy workers (data pool)
- cephfs_mirror_dir_pool_queue_size — directory queue capacity (bounded frontier)
- cephfs_mirror_file_pool_queue_size — max in-flight files (data pool cap)
- cephfs_mirror_sync_latest_snapshot — take the latest snapshot for syncing (skipping the intermediate snapshots)
- cephfs_mirror_remote_diff_base_upon_start — force remote diff base for comparison while start

Extra Stats Collection For the User

Command:

ceph fs snapshot mirror daemon status | jq


```
[
  {
    "daemon_id": 4275,
    "filesystems": [
      {
        "filesystem_id": 2,
        "name": "sourcefs1",
        "directory_count": 2,
        "peers": [
          {
            "uuid": "f87c1365-00cf-42b1-b69e-b5b401cb891a",
            "remote": {…
            },
            "stats": {
              "failure_count": 0,
              "recovery_count": 0,
              "hostname": "ceph-dev",
              "status": {
                "/ceph": {
                  "state": "syncing",
                  "current_syncing_snap": {
                    "id": 3,
                    "name": "snap_1",
                    "rfiles": 117328,
                    "rbytes": 3938815135,
                    "diff_base": "remote",
                    "files_in_flight": 5036,
                    "files_synced": 57769,
                    "files_deleted": 0,
                    "files_skipped": 0,
                    "files_bytes_synced": 2060234346,
                    "dir_created": 11058,
                    "dir_scanned": 11059,
                    "dir_deleted": 0,
                    "cache_hit": 0,
                    "time_elapsed": 14.311952318,
                    "transfer-rate": "1.072526Gbps"
                  },
                  "failure_count": 0,
                  "snaps_synced": 0,
                  "snaps_deleted": 0,
                  "snaps_renamed": 0
```

```
                  },
                  "failure_count": 0,
                  "snaps_synced": 0,
                  "snaps_deleted": 0,
                  "snaps_renamed": 0
                },
                "/linux": {
                  "state": "syncing",
                  "current_syncing_snap": {
                    "id": 4,
                    "name": "snap_1",
                    "rfiles": 91184,
                    "rbytes": 7781863729,
                    "diff_base": "remote",
                    "files_in_flight": 5036,
                    "files_synced": 21405,
                    "files_deleted": 0,
                    "files_skipped": 0,
                    "files_bytes_synced": 303953325,
                    "dir_created": 2777,
                    "dir_scanned": 2778,
                    "dir_deleted": 0,
                    "cache_hit": 0,
                    "time_elapsed": 4.251978513,
                    "transfer-rate": "545.388418Mbps"
                  },
                  "failure_count": 0,
                  "snaps_synced": 0,
                  "snaps_deleted": 0,
                  "snaps_renamed": 0
                }
              }
            }
          }
        ]
      }
    ]
  }
]
```

Thank you, any questions?