Improve RocksDB compaction in Ozone

Author: Wei-Chiu Chuang

Last update: 3/11/2025

Status: In Progress

We are seeing in several large production Ozone deployments where RocksDB seek is becoming a performance bottleneck due to consecutive tombstones.

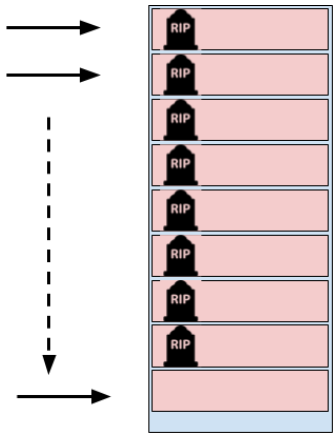
We need a solution to compact RocksDB and restore performance when it happens, ideally, without human intervention.

Symptoms

Users claim Ozone Manager becomes extremely slow to respond to any requests, leading to missing SLAs for workloads. A simple ‘ozone fs -ls’ command takes several minutes to return.

Jstack shows most time is spent in RocksDB iterator seek. Here are two examples: [1][2].

The root cause is inefficient iterator seek when there are consecutive tombstones due to many deletions. This is a known issue in the RocksDB community (also here and here).



Expected behavior

Ozone should be able to auto-compact RocksDB without human intervention, and performance should not cause noticeable impact afterwards.

Background context

Having too many L0 SST files degrades performance. Therefore RocksDB compacts them when the number of L0 files grows to 4.

When a RocksDB iterator seeks, it must skip tombstone keys (keys that are marked for dqeletion). Seek performance degrades when there are too many consecutive tombstone keys, which can happen when a large directory is deleted. It is possible to encounter this situation before the number of L0 files grows to 4 and triggers a compaction.

Workaround

> Before a permanent solution is implemented, the workaround is to shutdown a follower OM, RocksDB ldb tool to manually compact fileTable, restart the OM; repeat for the rest of OMs.

Solutions

Low-hanging fruits

1. Remove redundant seeks HDDS-12159

When a RocksDB iterator is initialized, it does a seekToFirst. Ozone would then do an explicit seek, therefore the seekToFirst becomes redundant. This is a low hanging fruit.

Status: fixed

2. Replace deletes with DeleteRange HDDS-12255

Replace a bunch of deletes with a single DeleteRange call. For example, deleting a directory would be a good use case since sub directories and files will have the same prefix. Since RocksDB 7.7.2, seek will skip to the end of a range delete, and maintain the performance.

3. Auto compaction upon deletion threshold HDDS-12254

RocksDB v8.11.3 and above supports a new feature to trigger compaction when the number of deletion tombstones exceeds a threshold in sst files. It will require a RocksDB upgrade because the current version is 7.7.3. However, it may not help with the tombstones in memtable.

4. Tunable compaction behavior HDDS-12173

There are several tunable compaction parameters. For example, the number of compaction/flush threads.

Medium size tasks

5. Offload block info from fileTable/keyTable

BlockInfo is bulky. Offloading it to a separate RocksDB column family will accommodate more keys in a single SST file and therefore reduce the number of SST files.

6. Heuristics to auto-compact

Monitor seek latency and trigger compaction if it exceeds a threshold.

- RocksDB supports time-based periodic compaction and TTL. Default is 30 days. We could make it tunable and set a smaller interval. See this too.

- Other heuristics
    - HDDS-12518 Compact RocksDB after consecutive deletions. Status: patch available.

7. Update to RocksDB 9
    a. Some of the bugs, improvements mentioned here are fixed in the newer versions. Consider updating to the latest RocksDB 9.10.0.
    Upgrade RocksDB version from 7 to 9
    b. Periodic_compaction_seconds compacts to the lowest level since 8.11.3

Command lines and metrics to help troubleshoot the issue

8. Trigger compaction on-demand

Develop an admin-only CLI to trigger compaction at OM at runtime.
HDDS-12310 Online repair command to perform compaction on om.db.
Status: patch available

9. RocksDB statistics

We should export the internal statistics to JMX. There is a small overhead so we may not want to enable it by default.

Update: this is already available behind a Ozone configuration property ozone.metastore.rocksdb.statistics = ALL or EXCEPT_DETAILED_TIMERS.

We are also developing a new Grafana dashboard to visualize the RocksDB stats: HDDS-12446

Status:patch available

Action Items

1. Reproduce the performance degradation using the internal lab environment;
2. decide which fixes to implement;
3. and use the internal repro to validate the fixes.

Appendix

[1] RocksDB iterator seek in rename

```
"OM StateMachine ApplyTransaction Thread - 0" #1411 daemon prio=5 os_prio=0 cpu=111238927.65ms elapsed=1475624.54s tid=0x00007f1acc403000
nid=0x172dbf runna

ble  [0x00007f02d0e54000]

   java.lang.Thread.State: RUNNABLE

        at org.rocksdb.RocksIterator.seekToFirst0(Native Method)

        at org.rocksdb.AbstractRocksIterator.seekToFirst(AbstractRocksIterator.java:48)

        at org.apache.hadoop.hdds.utils.db.RDBStoreAbstractIterator.seekToFirst(RDBStoreAbstractIterator.java:115)

        at org.apache.hadoop.hdds.utils.db.RDBStoreCodecBufferIterator.<init>(RDBStoreCodecBufferIterator.java:103)

        at org.apache.hadoop.hdds.utils.db.RDBTable.iterator(RDBTable.java:232)

        at org.apache.hadoop.hdds.utils.db.TypedTable.iterator(TypedTable.java:418)

        at org.apache.hadoop.hdds.utils.db.TypedTable.iterator(TypedTable.java:409)

        at org.apache.hadoop.hdds.utils.db.TypedTable.iterator(TypedTable.java:55)

        at org.apache.hadoop.ozone.om.request.file.OMFileRequest.checkSubFileExists(OMFileRequest.java:923)

        at org.apache.hadoop.ozone.om.request.file.OMFileRequest.hasChildren(OMFileRequest.java:846)

        at org.apache.hadoop.ozone.om.OzonePrefixPathImpl.<init>(OzonePrefixPathImpl.java:77)

        at org.apache.hadoop.ozone.om.request.OMClientRequest.checkACLsWithFSO(OMClientRequest.java:248)

        at org.apache.hadoop.ozone.om.request.key.OMKeyRenameRequestWithFSO.validateAndUpdateCache(OMKeyRenameRequestWithFSO.java:119)

        at org.apache.hadoop.ozone.protocolPB.OzoneManagerRequestHandler.handleWriteRequest(OzoneManagerRequestHandler.java:378)

        at org.apache.hadoop.ozone.om.ratis.OzoneManagerStateMachine.runCommand(OzoneManagerStateMachine.java:560)

        at org.apache.hadoop.ozone.om.ratis.OzoneManagerStateMachine.lambda$1(OzoneManagerStateMachine.java:353)

        at org.apache.hadoop.ozone.om.ratis.OzoneManagerStateMachine$$Lambda$1031/0x00007f02d191d500.get(Unknown Source)

        at java.util.concurrent.CompletableFuture$AsyncSupply.run(java.base@11.0.23/CompletableFuture.java:1700)

        at java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@11.0.23/ThreadPoolExecutor.java:1128)

        at java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@11.0.23/ThreadPoolExecutor.java:628)
```

```
        at java.lang.Thread.run(java.base@11.0.23/Thread.java:829)
```

[2] RocksDB iterator seek in directory delete

```
"KeyDeletingService#0" #1032 daemon prio=5 os_prio=0 cpu=1446151362.28ms elapsed=1475807.36s tid=0x00007f02eb029000 nid=0x1713bf runnable
[0x00007f02f69b80

00]

   java.lang.Thread.State: RUNNABLE

        at org.rocksdb.RocksIterator.seekDirect0(Native Method)

        at org.rocksdb.AbstractRocksIterator.seek(AbstractRocksIterator.java:73)

        at org.apache.hadoop.hdds.utils.db.RDBStoreCodecBufferIterator.seek0(RDBStoreCodecBufferIterator.java:125)

        at org.apache.hadoop.hdds.utils.db.RDBStoreCodecBufferIterator.seek0(RDBStoreCodecBufferIterator.java:30)

        at org.apache.hadoop.hdds.utils.db.RDBStoreAbstractIterator.seek(RDBStoreAbstractIterator.java:134)

        at org.apache.hadoop.hdds.utils.db.RDBStoreAbstractIterator.seek(RDBStoreAbstractIterator.java:33)

        at org.apache.hadoop.hdds.utils.db.TypedTable$RawIterator.seek(TypedTable.java:650)

        at org.apache.hadoop.hdds.utils.db.TypedTable$RawIterator.seek(TypedTable.java:619)

        at org.apache.hadoop.ozone.om.KeyManagerImpl.gatherSubDirsWithIterator(KeyManagerImpl.java:1884)

        at org.apache.hadoop.ozone.om.KeyManagerImpl.getPendingDeletionSubDirs(KeyManagerImpl.java:1871)

        at org.apache.hadoop.ozone.om.service.AbstractKeyDeletingService.prepareDeleteDirRequest(AbstractKeyDeletingService.java:355)

        at org.apache.hadoop.ozone.om.service.DirectoryDeletingService$DirDeletingTask.call(DirectoryDeletingService.java:180)

        at org.apache.hadoop.hdds.utils.BackgroundService$PeriodicalTask.lambda$run$0(BackgroundService.java:121)

        at org.apache.hadoop.hdds.utils.BackgroundService$PeriodicalTask$$Lambda$354/0x00007f02faa48cb0.run(Unknown Source)

        at java.util.concurrent.CompletableFuture$AsyncRun.run(java.base@11.0.23/CompletableFuture.java:1736)

        at java.util.concurrent.Executors$RunnableAdapter.call(java.base@11.0.23/Executors.java:515)

        at java.util.concurrent.FutureTask.run(java.base@11.0.23/FutureTask.java:264)

        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(java.base@11.0.23/ScheduledThreadPoolExecutor.java:304)

        at java.util.concurrent.ThreadPoolExecutor.runWorker(java.base@11.0.23/ThreadPoolExecutor.java:1128)

        at java.util.concurrent.ThreadPoolExecutor$Worker.run(java.base@11.0.23/ThreadPoolExecutor.java:628)

        at java.lang.Thread.run(java.base@11.0.23/Thread.java:829)
```