Dremio Software

Cleanup KV Store

Introduction

This document aims to explain the role of the KV Store in Dremio and introduce you to possible ways to clean up the KV Store as the size increases.

Importance of the KV Store

Any big data query engine (like Hive, Impala, Pig, Presto…etc.) that needs to run queries requires the datasets' metadata, which generates optimized query plans to process the data quickly with the least usage of resources (CPU/Memory). For that reason, these query engines must be configured with a metastore repository.

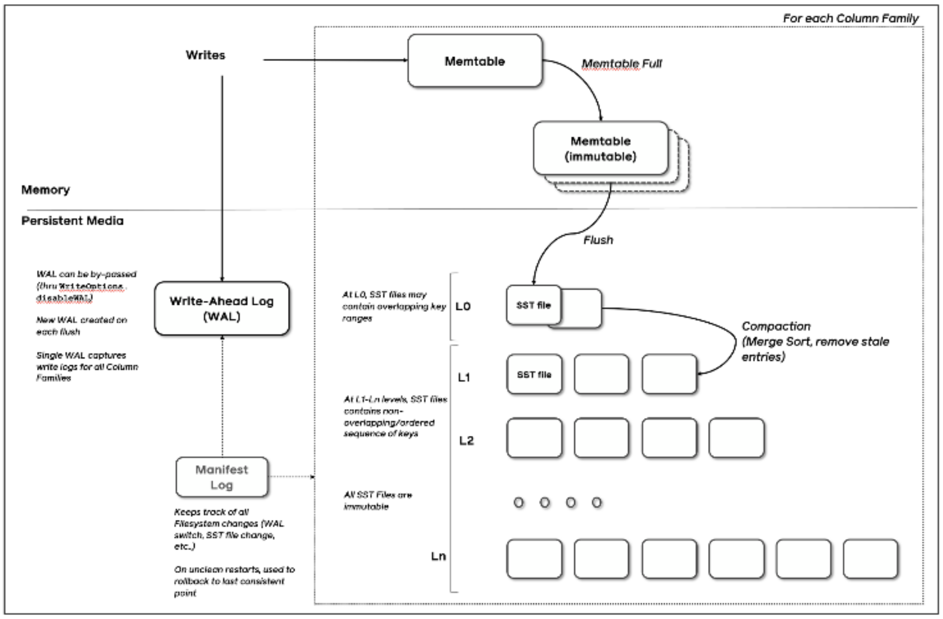
As a big data SQL engine, Dremio adopted KV Store (RocksDB) as a metastore to store the metadata of all source Physical Datasets (PDS). Dremio also has a “Semantic Layer” capability, enabling users to create Spaces/Folders/Virtual Data Sets (VDS) to organize the consumption with better data governance. These object definitions are also stored in the KV Store.

Note that Dremio stores the metadata in Iceberg format in the distributed storage for both Hive and Parquet file sources. Their references are stored in the KV Store which is needed in query planning.

KV Store Architecture

Dremio used RocksDB for the KV store inside Dremio. Facebook developed RocksDB using C++ to store Keys and Values which are arbitrary sized byte streams.

RocksDB organizes all data in sorted order and the common operations are Get(key), Newiterator(), Put(key, value), Delete(key), and SingleDelete(key). Basic constructs of RocksDB are memtable, sstfiles and logfile. Memtable is an in-memory data structure and the new writes are inserted into memtable and optionally written to loggfile. Logfile is also called a “Write Ahead Log” (WAL). All the transaction logs get stored in the logfile and if any restarts occur, it reprocesses all the transactions that were recorded in the logfile. When the memtable fills up, it's flushed to the sstfile on the storage and the corresponding log file can be safely deleted. The data in sstfile is sorted to facilitate easy lookup of keys.


For each Column Family
Writes
Memtable
Memtable Full
Memtable (immutable)
Memory
Persistent Media
Flush
WAL can be by-passed (thru WriteOptions.disableWAL)
New WAL created on each flush
Single WAL captures write logs for all Column Families
Write-Ahead Log (WAL)
At L0, SST files may contain overlapping key ranges
L0
SST file
Compaction (Merge Sort, remove stale entries)
L1
SST file
At L1-Ln levels, SST files contains non-overlapping/ordered sequence of keys
L2
Manifest Log
All SST Files are immutable
Keeps track of all Filesystem changes (WAL switch, SST file change, etc.)
On unclean restarts, used to rollback to last consistent point
Ln

Location of the KV Store

By default, this will reside at {$DREMIO_HOME}/data path. Administrators can customize this path in dremio.conf. It's required to use a location that is local to the coordinator as it makes the data read operations faster by the coordinator.

In the case of an HA (High Availability) setup, Dremio recommends using a Network Drive (NAS v4) as the KV Store location which can provide locking support with low latency read/write operations from the coordinators.

Expanding Size of the KV Store

As stated in earlier sections, Dremio holds the metadata of all its objects (PDS, VDS, Spaces, JOBs, Reflections..etc) in the KV Store. An increase in the KV Store size could occur for several reasons:

- Onboarding more Applications - KV Store holds the PDS (metadata like location, schema, partitions, splits ..etc), VDS, Reflections..etc. metadata info in the KV Store. So when there is an increase in the number of applications which in turn leads to adding Objects like PDS, VDS, Reflections..etc, it causes an increase in the KV Store Size.
- Increased workload - Dremio will hold job details and corresponding job profiles in the KV Store until its expiration. So if there is an increase in the workload, it will lead to an increase in KV Store size.
- Compaction Failures - Periodically Dremio merges small sstfiles into a larger sst file and it deletes key-value bindings (orphan files) that have been removed or overwritten. But for any reason, if the compaction process fails, it will increase the KV Store size.

Recommendations to Reduce KV Store Size

Dremio always advise customers to set up a monitoring framework (like Cloudwatch, Grafana..etc.) on the Disk space usage on which KV Store resides, to send alerts when the usage is more than the threshold (preference is set to 70%) and notifying the Dremio admins to investigate the reason for the increase in the usage and take the appropriate action either to bring down the usage or increase its size.

Though utilization is less than 70%, As a best practice Dremio recommends performing a regular cleanup activity (at least once every 6 months) when the size reaches 100GB to remove orphan files and compact the non-compacted files.

The command below shows the file system utilization stats on which the KV store resides.

```
df -h ${DREMIO_HOME}/data
```

As stated in the previous section, the increase in the KV Store size can be due to

- Onboarding of the new applications (OR)
- Workload increase (OR)
- Compaction failures

If the KV Store increase is due to the on-board of the new applications, Dremio always recommends increasing the Disk size on which the KV Store resides. For a graceful implementation of increasing the disk size and to be in a position to restore in case of unexpected issues, please follow the below steps.

- Stop the Dremio Service
- Take a backup of the KV Store
- Increase the disk size according to your own internally defined process
- Start the Dremio service

On the other hand, if the increase in KV Store size is due to an increase in the workload please implement the below:

- Reduce jobs max age - By default, Dremio holds the job details for 30 days. If it's viable for customers to reduce the age of jobs, Dremio recommends decreasing the age using the support key: jobs.max.age_in_days.
- Disabling verbose profiling - If the customer enables verbose profiling (planner.verbose_profile set to true) to get more granular level planning details, this will lead to occupying a good amount of KV Store size. We can obtain the profile size details by running the kvstore report (as shown in the section below). If it occupies more size and verbose is enabled, Dremio always recommends disabling the verbose profiling (planner.verbose_profile set to false). Enabling Verbose profiling is only advisable if the recommendation comes from the support team as part of its debugging for any query failures or performance issues. Once the debugging activity is completed, make sure to disable the verbose profiling.

In the case of Dremio failures in the periodic compaction, we will start to see a huge number of sst files accumulated in {$DREMIO_HOME}/data path and the number will keep growing. If this is the case, Dremio recommends implementing the below immediately.

-
- **Compact KV Store** - using the below demo-admin command we can do KV Store compaction which can merge small sst files into larger sst files.

  Sudo -u dremio /opt/dremio/bin/dremio-admin clean -c

- **Deleting orphan files** - there are some cases where files reside in the KV Store, though they have expired or have no references to them. Run the below demo-admin command to remove orphan files.

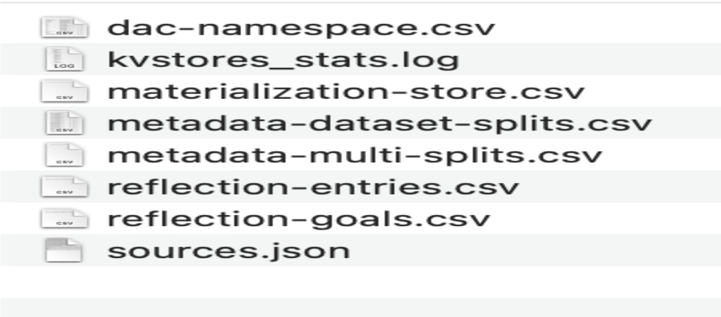
  Sudo -u dremio /opt/dremio/bin/dremio-admin clean -o

Generating KV Store Report

To track the storage occupied by various Dremio components in RocksDB, we can use the below REST API call.

```
curl --location --request GET '<<dremio_host>>:9047/apiv2/kvstore/report?store=none' \
--header 'Authorization: _dremio<<dremio_token>>' > kvstore_summary.zip
```

Once you issue the command, as shown below various statistics files get created and they are zipped into kvtsore_summary.zip.



dac-namespace.csv
kvstores_stats.log
materialization-store.csv
metadata-dataset-splits.csv
metadata-multi-splits.csv
reflection-entries.csv
reflection-goals.csv
sources.json

kvstore_stats.log is the file that provides the summary of storage occupancy/number of keys by each type of Dremio object.

The report data looks as described below:

```
profiles
      basic rocks store stats
            * Estimated Number of Keys: 9051
            * Estimated Live Data Size: 148734170
            * Total SST files size: 148734170
            * Pending Compaction Bytes: 0
            * Estimated Blob Count: 0
            * Estimated Blob Bytes: 0
```

```
jobs
      basic rocks store stats
            * Estimated Number of Keys: 11003
            * Estimated Live Data Size: 3893686
            * Total SST files size: 7210837
            * Pending Compaction Bytes: 0
            * Estimated Blob Count: 0
            * Estimated Blob Bytes: 0

      Index Stats
            * live records: 9051
            * deleted records: 240
```

```
commit_log
      basic rocks store stats
            * Estimated Number of Keys: 1765
            * Estimated Live Data Size: 2669008
            * Total SST files size: 2669008
            * Pending Compaction Bytes: 0
            * Estimated Blob Count: 0
            * Estimated Blob Bytes: 0
```

```
datasetVersions
      basic rocks store stats
            * Estimated Number of Keys: 948
            * Estimated Live Data Size: 1855236
            * Total SST files size: 1907324
            * Pending Compaction Bytes: 0
            * Estimated Blob Count: 0
            * Estimated Blob Bytes: 0
```

```
metadata-multi-splits
      basic rocks store stats
            * Estimated Number of Keys: 0
            * Estimated Live Data Size: 6643
            * Total SST files size: 22004
            * Pending Compaction Bytes: 0
            * Estimated Blob Count: 0
            * Estimated Blob Bytes: 0
```

The report clearly states how much size (Total SST file size) is occupied by each section and how many bytes are non-compacted (Pending Compaction Bytes).

If the disk utilization is more than a threshold and the report shows more “Pending Compaction bytes” under one or more sections, then initiate the compaction process immediately.

Command to use:

```
sudo -u dremio /opt/dremio/bin/dremio-admin clean -c
```

If the usage (Total SST files size) by jobs section is more, then revisit the settings of jobs.max.age_in_days and if it's viable to decrease its value, Dremio recommends decreasing it.

Below is the command to delete jobs and their associated profiles that are older than the specified threshold days.

Command to use:

```
sudo -u dremio /opt/dremio/bin/dremio-admin clean --max-job-days=7
```

If the usage (Total SST files size) is more in the profiles section, then as stated earlier please consider disabling verbose profiling (planner.verbose_profile set to false) in case it's enabled.