Separation Is for Better Reunion: Data Lake Storage at Huawei

ABSTRACT

Abstract here.

PVLDB Reference Format:
. Separation Is for Better Reunion: Data Lake Storage at Huawei. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

Intro here.

2 MOTIVATION

3 ARCHITECTURE

4 STREAM AND TABLE STORAGE OBJECT

In this section, we introduce the stream object and table object, purpose-built storage abstractions designed for efficient storage and access of stream and table data in the storage layer.

4.1 Stream Object

The stream object is a storage abstraction in the store layer that efficiently supports key-value message streaming at scale. It stores a partition[1] of key-value pairs for continuous message streams, organized as collections[2] of data slices. Each slice can contain up to 256 records as depicted in Figure 3. Incoming message records are appended to a specific slice in a stream object based on its topic, key, and offset.

Stream objects operations. The stream object operates similarly to the block and file storage abstractions, providing read and write functionality for stream storage. Figure 2 outlines key operations supported by the stream object, including creating and destroying a stream object with functions `CreateServerStreamObject` (line 1-3) and `DestroyServerStreamObject` (line 5-6) respectively. The `*option` field (line 2) sets storage configurations, such as data redundancy methods (replicate or erasure code) and I/O quotas, so as to ensure enterprise-level reliability and performance. The assigned `objectId` (line 3) serves as a unique identifier for operating the stream object. The `AppendServerStreamObject` function appends incoming records[3] to the stream object and returns the starting offset of the appended records. The `ReadServerStreamObject` function reads the stream object starting from a specified offset, with control conditions such as the length of the read specified in the `readCtrl` field. Since the message service is designed to support real-time streaming, it is configured to return all subsequent messages[4] unless specified or reaching quota limits.[5] `IO_CONTENT_S` (line 10 and 17) is a data structure that provides non-blocking I/O by using buffers to enhance the performance of both writing and reading operations.

Write stream messages. We discuss how to write messages into `StreamLake`and endure enterprise-level load-balanced and redundant persistence for the stream objects, which is achieved on the basis of SSD and HDD storage pools. As shown in Figure 3, the messages are first assigned to stream object slices based on topics, keys, and offsets (Figure 3-a,b,c).[6] Then, a distributed hash table is leveraged to ensure even data distribution for load-balance storage (Figure 3-d). Specifically, data slices will be distributed evenly to 4096 logical shards, each of which has the storage space managed by persistence logs (PLog, Figure 3-e)[7]. Each PLog unit is a collection of persistence services in OceanStor [] that controls a fixed amount of storage space on multiple disks and provides 128 MB of addresses per shard. When a message is received, the PLog unit replicates it to multiple disks for redundancy (Figure 3-f). Key-value databases[8] serve as indexes for PLogs for fast record lookup.

4.2 Table Object

We also extend the storage object layer in StreamLake to support table-like[9] operations for more effective data storage and management, similar to lakehouses [][10]. The table storage uses an open lakehouse format[11] with optimizations[12] for faster metadata access. The table abstraction is logically defined by a directory of data and metadata files, as shown an example in Figure 4.

`Data` directory. Table objects are stored in Parquet files of the `data` directory. In this example, the table is partitioned based on the date column, so the data objects are separated into different sub-directories by date. Each sub-directory name represents its partition range[13]. The data objects in each Parquet file are organized as row-groups[14] and stored in a columnar format for efficient data analysis. Footers in the Parquet files contain statistics to support data skipping within the file[15].

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

[1] – one partition or partitions
[2] – a collection or collections
[3] – an incoming record?
[4] *why?
[5] *The context of this sentence is not clear.
[6] @ how dies it work?
[7] –Plog 4096
[8] –can we name it specifically?
[9] – a term?
[10] * "similar to lakehouse" is weird
[11] –a term? what does it mean?
[12] –what? a part in Fig4?
[13] –partition key?
[14] –a term?
[15] –remove within the file?

Metadata directory keeps track of the table schema, file addresses of the table, its partitions and transaction commits etc.[16], which are organized into three levels: commit, snapshot, and catalog, as shown in Figure 4-(b, c, d).

Commits are Arvo files that contain file-level[17] metadata and statistics such as file paths[18], record counts, and value ranges for the data objects. Each data insert, update, and delete operation will generate a new commit file to record changes to the data object files[19].

Snapshots are index files that index valid commit files for a specified time period. These snapshots document commit statistics such as current file and row count, added files and rows, and removed files and rows[20] as data operation logs. Along with commits, snapshots provide snapshot-level isolation to support optimistic concurrency control. Readers can access the data by reading from the valid commit files, while changes made by a writer will not be visible to readers until they are committed and recorded in a snapshot. This allows for multiple readers and one writer to access the data simultaneously without the need for locks.

Snapshots also monitor the expiration of all commits, making them essential for supporting time travel. Time travel queries allow data to be viewed as it appeared at a specific time[21]. By keeping old commits and snapshots, the table object enables the use of a timestamp to look up the corresponding snapshot and commits, so as to access to historical data.

Catalog describes the table object, including the profile data such as the table ID, directory paths, schema, snapshot descriptions, modification timestamps, etc. The data and metadata files are stored in the table directory[22], except for the catalog, which is stored in a distributed key-value engine [23] optimized for RDMA and SCM to ensure fast metadata access. The data and metadata files are converted to PLogs in the underlying storage for redundant persistence as discussed above.

[16] *how to correspond to the three levels? and the following illustration order?
[17] –a term?
[18] @directory path
[19] –data object files or data object?
[20] –messy
[21] @
[22] @
[23] –can we name it? or cite