Iceberg Catalog as a Service
Agenda

• Apache Iceberg and Catalogs
• History of Hive Metastore
• REST Catalog Highlights
• Choosing the Right Catalog
What is Iceberg?

Iceberg is a high-performance format for huge analytic tables. Iceberg brings the reliability and simplicity of SQL tables to big data, while making it possible for engines like Spark, Trino, Flink, Presto, Hive and Impala to safely work with the same tables, at the same time.
Catalog

Where are all my tables?

How can I access them (safely)?
Catalogs Supported in Apache Iceberg

Popular Choice

- HiveCatalog (incubation/2018)
- HadoopCatalog (Nov 2019)
- JDBCCatalog (June 2021)
- RESTCatalog (May 2022)

With Vendor Support

- GlueCatalog
- SnowflakeCatalog
- You can even build your own catalog
History of Hive Metastore

• Apache Hive was introduced as warehousing solution over map-reduce framework back in VLDB 2009

• Hive metastore was included as a system catalog from Hive project, used to keep track metadata of tables, such as schema, key-value properties and ownership.

• Most Iceberg users migrated from Hive, can reuse the same hive metastore for catalogue
Hive Locking Problem

1. Lock Table
2. Get Table
3. Alter Table
4. Unlock Table

Iceberg Table
Commit
to HiveCatalog
Hive Locking Problem

![Diagram showing the steps: Lock Table, Get Table, Alter Table, Unlock Table. Highlighted step: Unlock Table is red X.]

WARN Tasks: Retrying task after failure: Waiting for lock.

org.apache.iceberg.hive.HiveTableOperations$WaitingForLockException: Waiting for lock.
Caused by: org.apache.iceberg.exceptions.CommitFailedException: Timed out after 182592 ms waiting for lock on namespace.table
Path to Lock Free

Lock-free implementation iff

• Upgrade Hive metastore server with fix HIVE-26882
• Upgrade all Iceberg library in engines to 1.3
• All engines need to disable Hive locks on commit
• 🔥 Risk of corrupting table if handled incorrectly

https://github.com/apache/iceberg/pull/6570
We can solve any problem by introducing an extra level of indirection - Andrew Koenig
Iceberg REST Catalog APIs

//Namespaces API
POST /v1/{prefix}/namespaces
GET /v1/{prefix}/namespaces
GET /v1/{prefix}/namespaces/{ns}
POST /v1/{prefix}/namespaces/{ns}/properties
DELETE /v1/{prefix}/namespaces/{ns}

//Configuration API
GET /v1/config

//Authorization API
POST /v1/oauth/tokens
Iceberg REST Catalog APIs

//Tables API
POST  /v1/{prefix}/namespaces/{ns}/tables
POST  /v1/{prefix}/namespaces/{ns}/register
GET   /v1/{prefix}/namespaces/{ns}/tables/
GET   /v1/{prefix}/namespaces/{ns}/tables/{tbl}
POST  /v1/{prefix}/namespaces/{ns}/tables/{tbl}
DELETE /v1/{prefix}/namespaces/{ns}/tables/{tbl}
HEAD  /v1/{prefix}/namespaces/{ns}/tables/{tbl}
POST  /v1/{prefix}/tables/renames

//Metrics API
POST  /v1/{prefix}/namespaces/{ns}/tables/{tbl}/metrics
Choosing the Right Catalog

- Language agnostic implementation
- Pluggable access control
- Aggregated metrics report
- Support of Hive Tables
Iceberg Commit Metrics

```sql
insert into iceberg.foo.bar
values (…)
```

POST /v1/prefix/namespaces/foo/tables/bar/metrics

table-name: iceberg.foo.bar

operation: append

metrics:

- added-data-files: {unit: count, value: 5}
- added-files-size-bytes: {unit: bytes, value: 3323}
- added-records: {unit: count, value: 5}
- attempts: {unit: count, value: 1}

... total-duration: {count: 1, time-unit: nanoseconds, total-duration: 270419834}

metadata: {app-id: local-1695675112222, engine-name: spark, engine-version: 3.3.3,
Iceberg Scan Metrics

POST /v1/prefix/namespaces/foo/tables/bar/metrics

table-name: iceberg.foo.bar

filter: {term: id, type: gt-eq, value: (1-digit-int)}

metrics:

result-data-files: {unit: count, value: 3}

scanned-data-manifests: {unit: count, value: 1}

skipped-data-files: {unit: count, value: 2}

skipped-data-manifests: {unit: count, value: 0}

...  

total-planning-duration: {count: 1, time-unit: nanoseconds, total-duration: 37548625}

metadata: {app-id: local-1695675112222, engine-name: spark, engine-version: 3.3.3,}
Migrate Catalog

Delegate
Set up REST catalog endpoints and delegate all requests to original HiveCatalog

Switch
Update engine configuration (Spark/ Flink/Trino) so it connects to REST instead of Hive
Migrate Backend

Prepare
Provision new relational database for JDBC backend and restrict network access

Migrate
Leverage register-table API to migrate Iceberg tables from Hive to JDBC backend
Performance Under Load
Performance Breakdown by API

Create Table
- Median: 737 ms
- Minimum: 158 ms

Drop Table
- Median: 524 ms
- Minimum: 151 ms

Get Table
- Median: 151 ms
- Minimum: 72 ms

Update Table
- Median: 722 ms
- Minimum: 165 ms
Benchmark Setup

• Server
  • 2 REST endpoints on Kubernetes
  • 3 pods, 1 core and 4GiB memory for each

• Dependencies
  • Iceberg 1.2.1
  • Hive metastore 3.1
  • PostgreSQL 15

• Clients
  • Apache JMeter to simulate client requests
Contribute Back to Community

- OpenAPI: Add namespaceExist API: #8569
- Core: Extend ResolvingFileIO to support BulkOperations: #7976
- Build: Add openapi label: #7721
- OpenAPI: TableRequirement definition and parser mismatch: #7700
- Core: Fix SetDefaultPartitionSpec to use specId instead of schemaid #7350
- OpenAPI: Return 204 on no content response #7229
- OpenAPI: Correct snapshot id and time ms int format #6921
Thanks For Attending

linkedin.com/in/hongyue-zhang-3abb7378

@dramaticlly