CLOUDERA

Hive Iceberg Integration

June 2022

Peter Vary

AGENDA

Hive overview

Iceberg overview

Iceberg integration points

Hive 4.0.0-alpha-1 & future plans

Hive is an SQL engine

Hive is an SQL engine which runs queries above Petabyte scale data

"The Apache Hive ™ **data warehouse** software facilitates reading, writing, and managing large datasets residing in **distributed** storage using **SQL**."

- Open Sourced by Facebook in 2010, continuously improved by the community
  - From HiveCli - monolithic command line tool
  - To Cloud native
    - Client: submitting queries - JDBC / ODBC / BeeLine
    - Compiler: query compilation and optimization - HiveServer2
    - **Metastore**: storing and serving table and partition metadata - HiveMetastore
    - Executor: managing query execution - TezAM / LLAP
- **Partition**: Part of the table based on one or multiple columns

AGENDA

Hive overview

Iceberg overview

Iceberg integration points

Hive 4.0.0-alpha-1 & future plans

What is Iceberg?

Iceberg is a library which execution engines can invoke to create/read/write tables

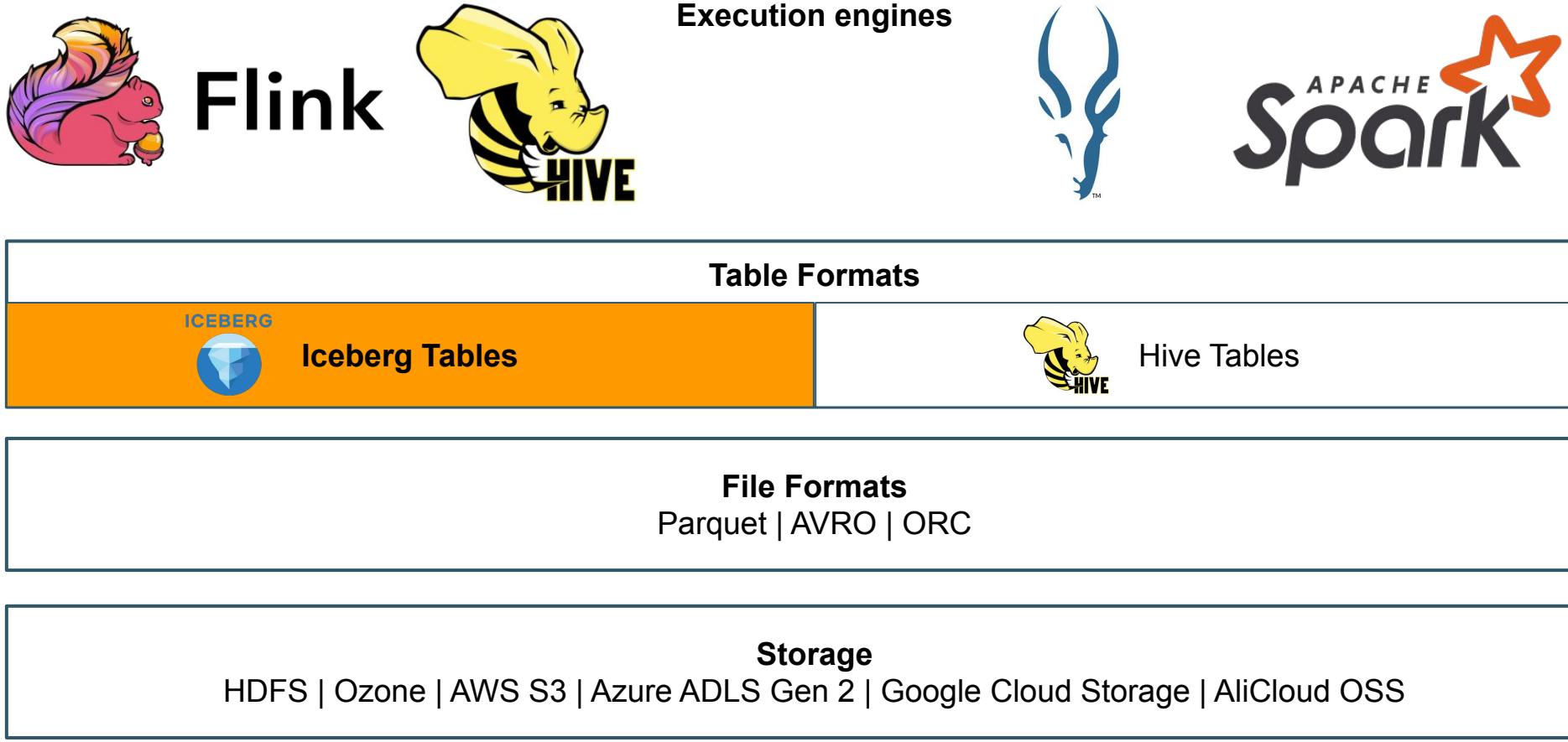
"Apache Iceberg is a new **table format** for storing **large**, **slow-moving** tabular data. It is designed to improve on the de-facto standard table layout built into Hive, Trino, and Spark."

"Apache Iceberg is an open **table format** for **huge analytic datasets**. Iceberg adds tables to compute engines including Spark, Trino, PrestoDB, Flink and Hive using a high-performance table format that works just like a SQL table."

- Large: could be as high as PBs
- Slow-moving: should not change more than few times per minute
- Library which provides an API to manipulate and read tables

Iceberg is a table format

Data storage layers



Hive format vs Iceberg format

Iceberg tracks table state on the file-level

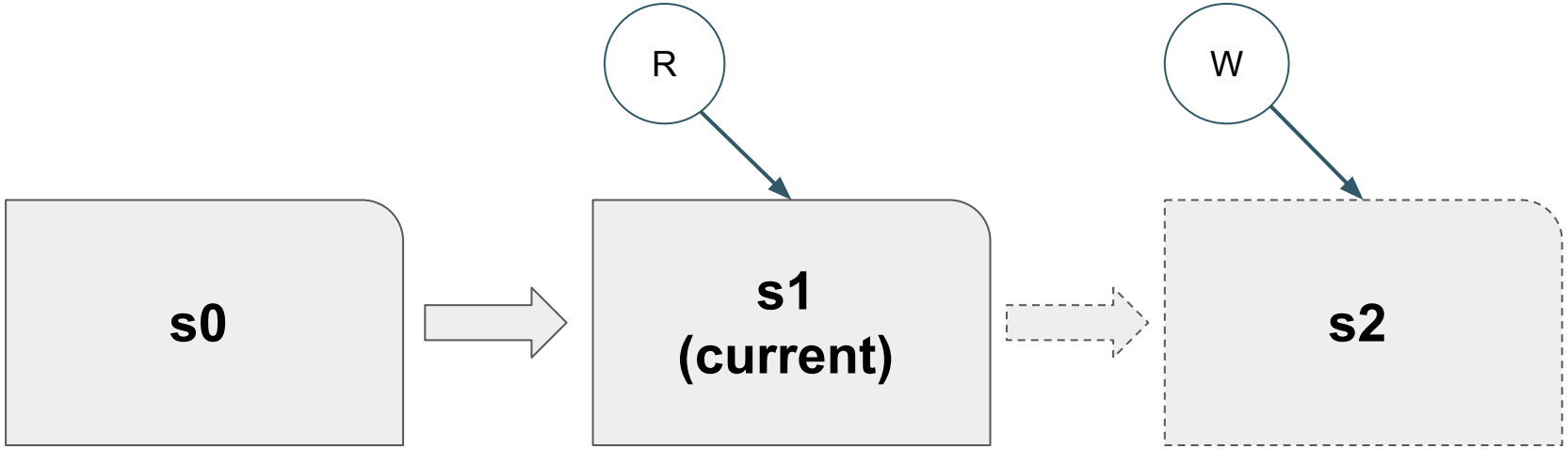
- Tables / partitions and their metadata are stored in HMS
- Keeps track of data on the **directory-level**

  /path/to/table/part_key=part_val1/*
  /path/to/table/part_key=part_val2/*
- Need to list dirs to get data files
- Information is stored in two places: HMS and the File System
- **Non versioned** metadata

- Uses metadata files (in the FS) to:
  - track data files
  - store partition information & metrics about data files
- Keeps track of data on the **file-level**
- The table knows which exact data files belong / belonged to the table at any given point in its history
- **Versioned** data and metadata

Iceberg has versioned metadata

Iceberg writes create new immutable snapshots to form a linear history



- Each table update (write/alter) produces a new **snapshot** -> linear history
- Readers always pin down a certain snapshot at the beginning of execution
- Writers produce new snapshots in isolation, then commit

Iceberg metadata structure

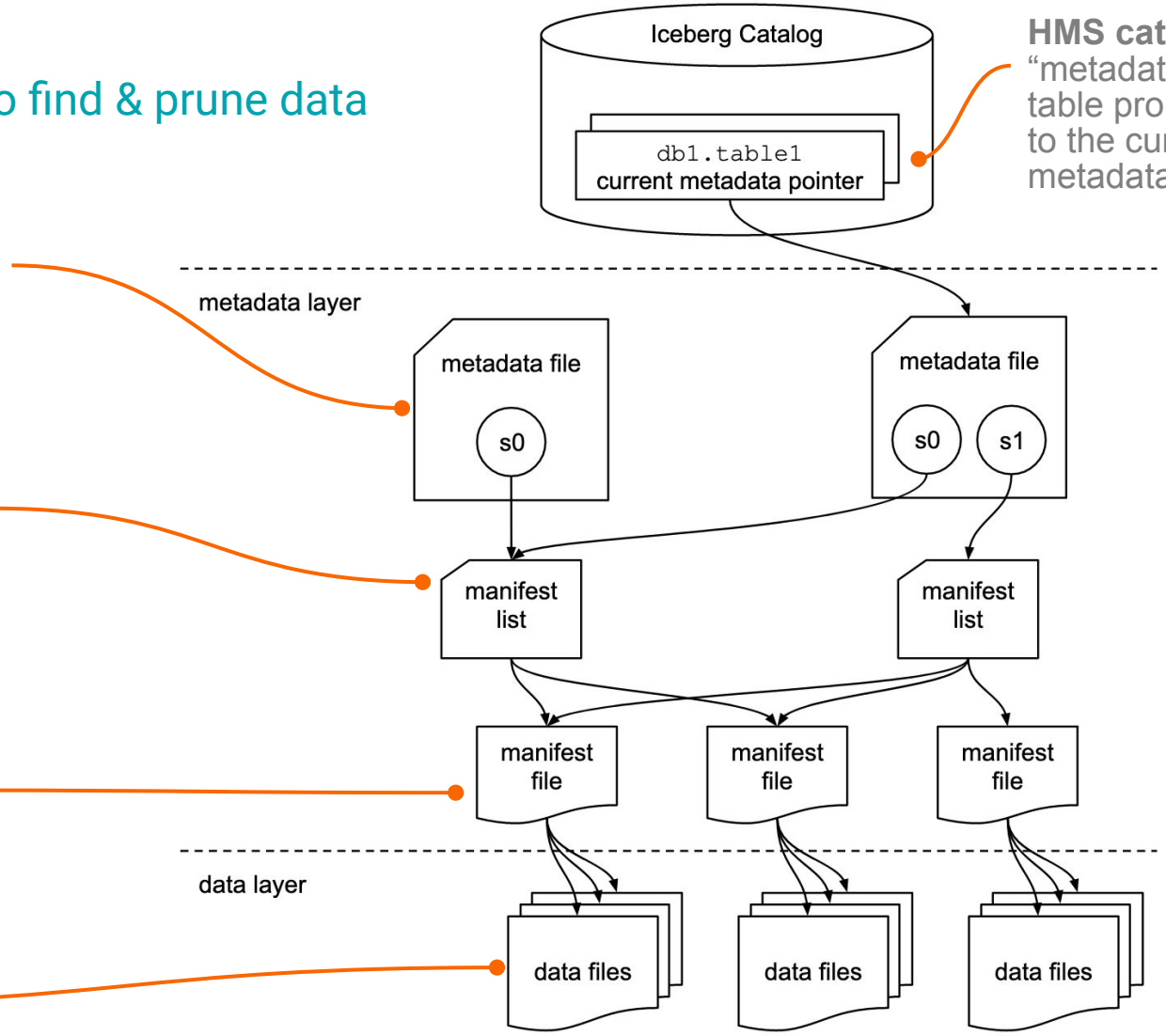
Iceberg uses multiple layers of metadata files to find & prune data

Metadata file:
- Contains the schema, partition spec, the history of the table, and the current snapshot

Manifest list:
- Tracks all manifest files by partition ranges
- Can prune entire manifests during planning based on partition value

Manifest file:
- Tracks data files across many partitions
- Stores partition info and column metrics for each data file (for pruning and optim.)

Data files (Parquet/ORC/Avro)



Iceberg main benefits

Iceberg brings new features and addresses many shortcomings of Hive tables

- Isolated reads and writes (snapshot isolation) without locking
- Time travel (based on a specific date / snapshot ID)
- File-level tracking -> No expensive S3 dir listings
- No central metastore bottleneck (better scaling, more granular partitioning)
- Partition evolution (w/o table rewrite)
- Partition transform functions (truncate(), bucket(), year(), month(), etc.)
- Reliable schema evolution across all 3 formats (no surprises!)

Iceberg write concurrency - known limitation

Iceberg commit operation is sensitive for high write concurrency

- Iceberg uses optimistic locking for writes
  - On write conflict, the slower process must recreate the new metadata file
- Iceberg serializes commits on **table-level** (via exclusive write lock), not partition-level
- Expected to be **slower for concurrent writes** than partitioned Hive ACID tables

AGENDA

Hive overview

Iceberg overview

Iceberg integration points

Hive 4.0.0-alpha-1 & future plans

Iceberg integration

Hive uses HadoopFileIO implementation

FileIO (API for handling files)

- Write - Create a new file and write to it
- Read - Open an existing file and read from it
- Delete - Delete an existing file

Different implementations (S3, GCS, Ecs, Local, HadoopFileIO)

Hive uses HadoopFileIO

- Generic layer above the supported FS implementations

Iceberg integration

Hive uses Iceberg HiveCatalog implementation

Catalog (API to switch snapshot):

- Commit - Change should be atomic

Hive recommends HiveCatalog, since Hive and Iceberg table properties are in sync

HiveCatalog:

- New table properties: *metadata_location=s3a://…*
- Iceberg writers commit by swapping the *metadata_location* pointer
- Lock is used, using a table-level Hive Metastore lock

Hive specific Iceberg integration

Hive enables reading / writing Iceberg tables via a new storage handler implementation

HS2/execution:

- HiveIcebergStorageHandler

```
CREATE TABLE t (i int, ts timestamp) STORED BY ICEBERG;
```

- HiveIcebergInputFormat - Implements Hive reads
- HiveIcebergOutputFormat - Implements Hive writes
- HiveIcebergMetaHook - Implements changes needed for DDL operations
- Hive-QL enrichments: new syntax for partition specification, time travel, etc.

AGENDA

Hive overview

Iceberg overview

Iceberg integration points

Hive 4.0.0-alpha-1 & future plans

Hive 4.0.0-alpha-1 is released

Contains the new Iceberg table integration

- After 2 years we finally released the alpha version of Hive 4.0.0
- New features:
  - Cloud perf. optimizations (tables on S3)
  - Transaction/locking perf. optimizations
  - Scheduled queries
  - Datasketches for statistics
  - Iceberg integration
- Planned changes before 4.0.0 release (subject to change):
  - Java upgrade
  - Hadoop 3.3 upgrade
  - Iceberg V2 (Delete / Update / Merge)

Hive Iceberg integration

SQL syntax changes

- Create table

```
CREATE TABLE t (i int, ts timestamp)
    PARTITIONED BY SPEC (month(ts), bucket(2, i)) STORED BY ICEBERG;
```

- Migrate table

```
ALTER TABLE t SET TBLPROPERTIES
    ('storage_handler'='org.apache.iceberg.mr.hive.HiveIcebergStorageHandler');
```

- Iceberg metadata table (history, partitions, files, etc) queries

```
SELECT * FROM default.t.history;
```

- Time travel

```
SELECT * FROM t FOR SYSTEM_TIME AS OF '2021-08-09 10:35:57';
SELECT * FROM t FOR SYSTEM_VERSION AS OF 1234567890;
```

What's on the horizon for Iceberg / Hive?

Further tasks

Task	Description	Status
Data mutation	Delete / Update / Merge (in progress)	Iceberg: ready Hive: in progress
Optimizations	Vectorization / Caching / Query optimizations	Hive: in progress
Statistics	Use Iceberg column statistics	Iceberg: in progress
Table optimization / compaction	Iceberg community implementing this using Spark	Iceberg: ready Hive: not started yet
Encryption	For security	Iceberg: ready Hive: not started yet
Multi table transactions	Maybe even handling multi table transactions with ACID tables	Iceberg: API draft
Data sharing	By creating secure views above specific snapshots	Iceberg: ready Hive: not started yet

Useful bits

Links and tips

- https://iceberg.apache.org/
- https://iceberg.apache.org/docs/latest/hive/
- https://github.com/apache/iceberg/pull/4915 - new Hive doc in progress

- Docker image

```
# Download and start HiveServer2

docker run --rm -p 10000:10000 --name hive4 -e HIVE_VERSION=4.0.0-alpha-1 -e
TEZ_VERSION=0.10.1 -v hive-dev-box_work:/work kgyrtkirk/hive-dev-box:bazaar

# Connect to HiveServer2 - in another shell after the HS2 has been started

docker exec -it hive4 /bin/bash --login -e safe_bl
```

THANK YOU!

And we are hiring :)