Hudi Connector Support

openLooKeng

https://openlookeng.io

Content

Apache Hudi Overview

Hudi Tables and Queries

Hudi Connector Support in OpenLooKeng

Live Demo

Apache Hudi Overview

- Hudi [1]: Uber Engineering’s Incremental Processing Framework on Apache Hadoop -- 2017
- Apache Hudi (Apache Software Foundation Top Level Project) – 2019
- Motivation

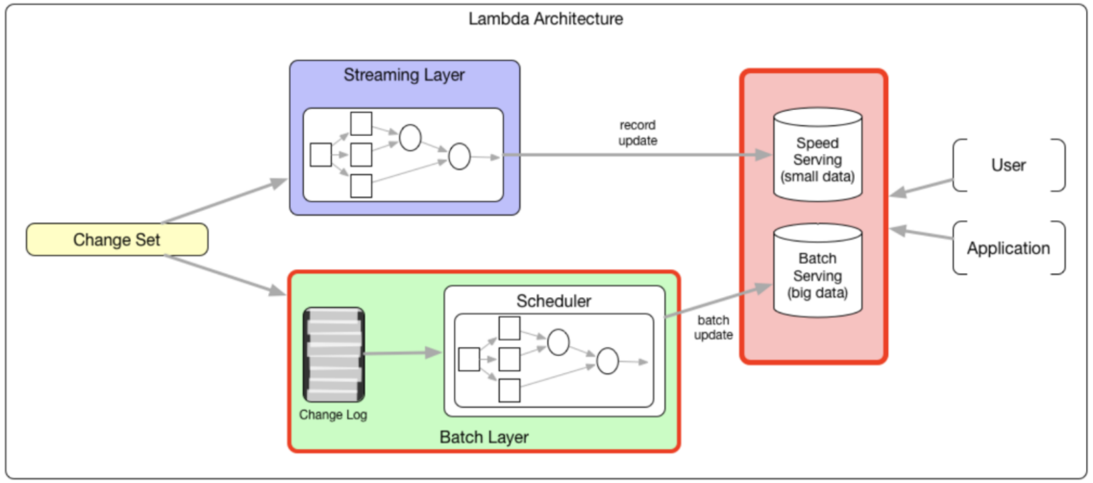


Figure 1: Lambda architecture requires double compute and double serving.

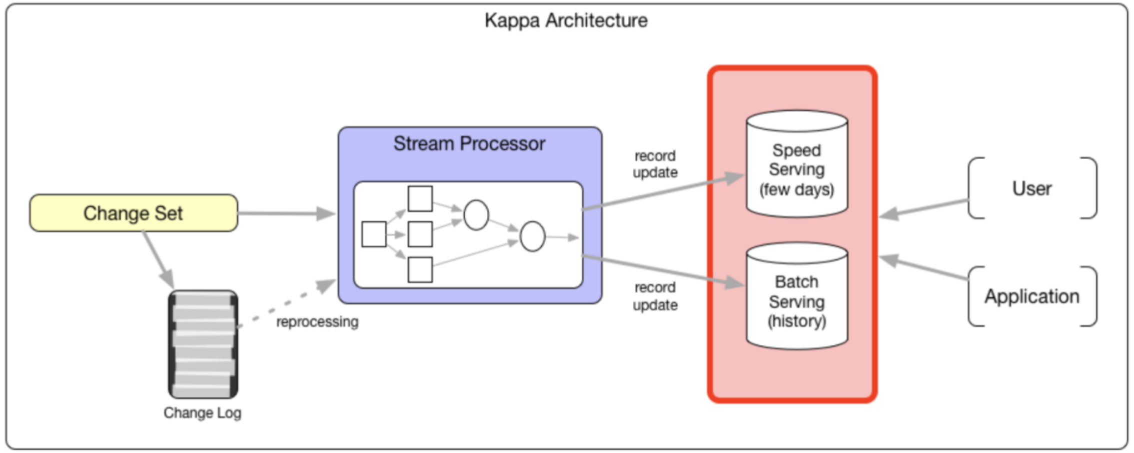


Figure 2: Kappa architecture simplifies computing by unifying processing, but serving complexity still exists.

- **Lambda architecture** is a common data processing architecture that proposes double compute with streaming and batch layer.
- **Kappa architecture** argues that a stream processing engine could be a general-purpose solution for computations

openLooKeng
https://openlookeng.io

Apache Hudi Overview

- This fundamental tradeoff between **data ingest latency**, scan performance, and **compute resources** is unavoidable

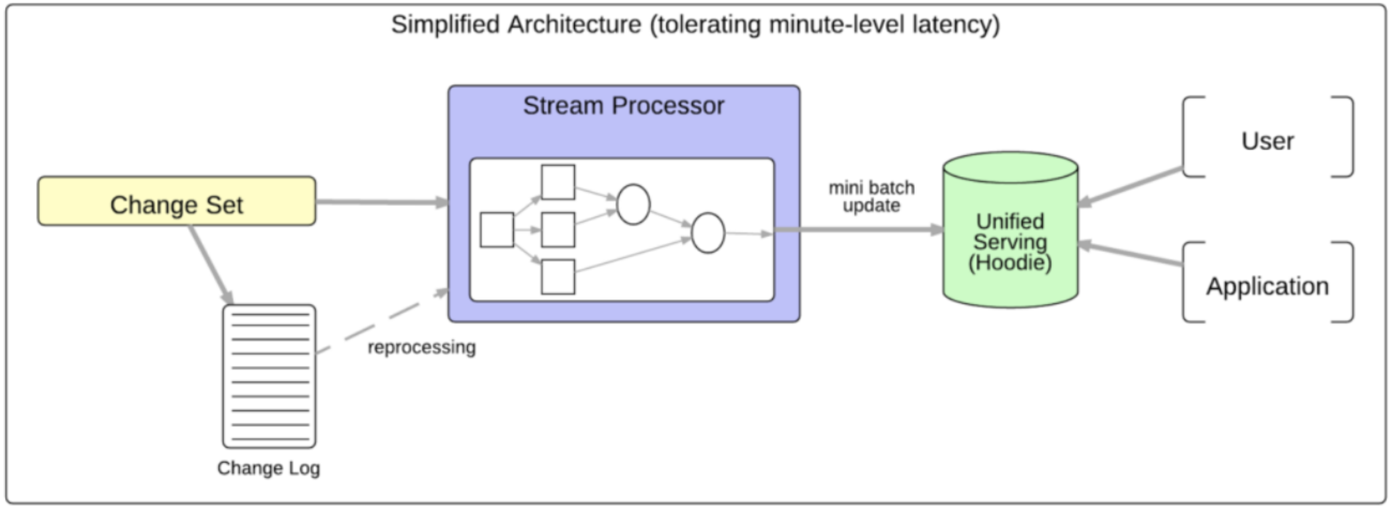
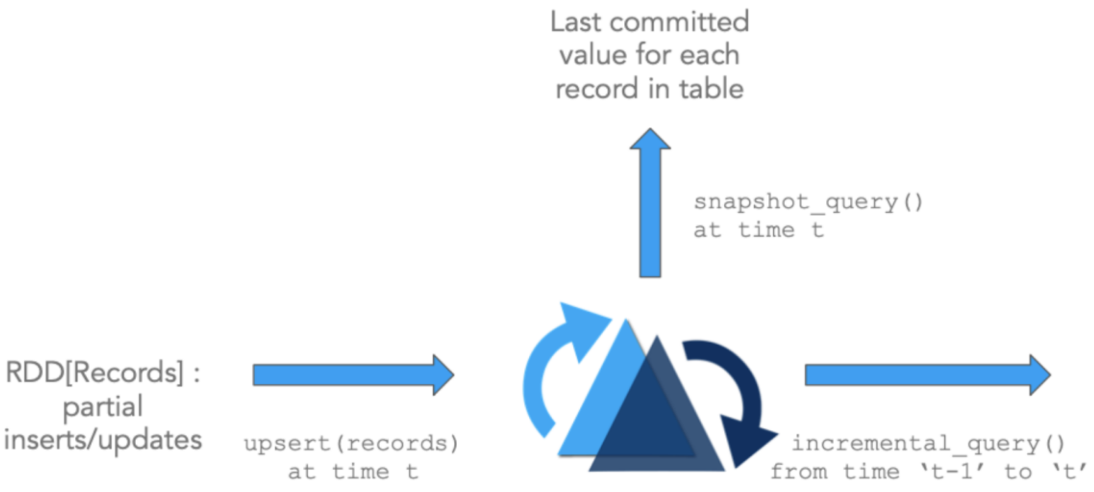


Figure 3: Hudi simplifies serving for workloads tolerating minute-level latency.

- Ability to quickly apply mutations to large HDFS datasets
- Data storage options that are optimized for analytical scans (columnar file formats)
- Ability to chain and propagate updates efficiently to modeled datasets

Apache Hudi Overview

- Apache Hudi (**H**adoop **U**psert **D**elete and **I**ncremental) is a fast growing data lake storage system
  - introduces primitives such as upserts, deletes and incremental queries
  - stores on the Hadoop Distributed File System (HDFS) or cloud stores
  - integrates well with popular query engines such as Hive, Spark, Impala
- Hudi enables stream processing in addition to typical batch-processing relying on two new primitives
  - **Update/Delete Records:** fine grained file/record level indexes; transactional guarantees
  - **Change Streams:** provides first-class support for obtaining an incremental stream

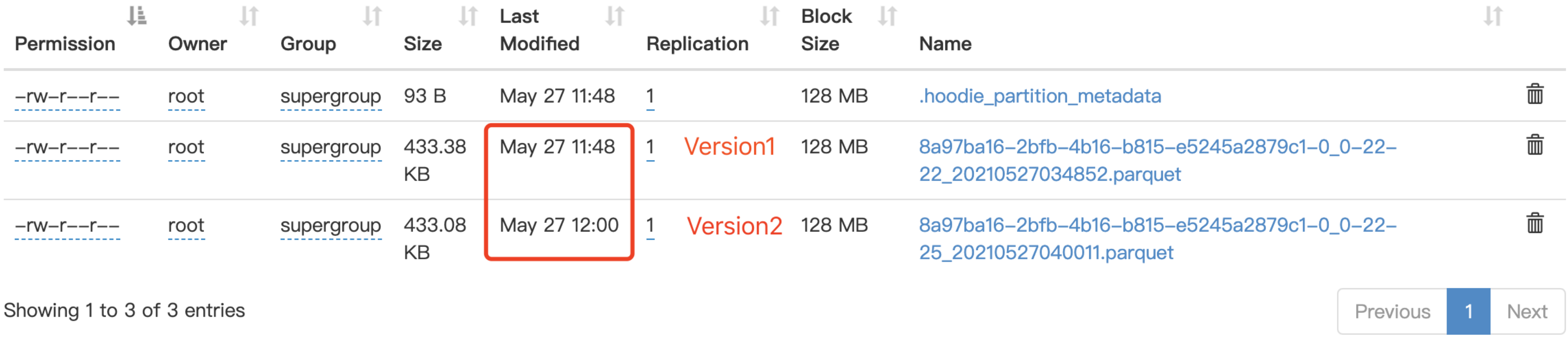


Last committed value for each record in table
snapshot_query() at time t
RDD[Records] : partial inserts/updates
upsert(records) at time t
incremental_query() from time 't-1' to 't'
All records updated/created in time range

https://openlookeng.io

Hudi Tables and Queries

- Table Types
  - **Copy On Write (COW):** Stores data using exclusively **columnar** file formats (e.g parquet). Updates version & rewrites the files by performing a synchronous merge during write.
  - **Merge On Read (MOR):** Stores data using file versions with combination of **columnar** (e.g parquet) + **row** based (e.g avro) file formats. Updates are logged to delta files



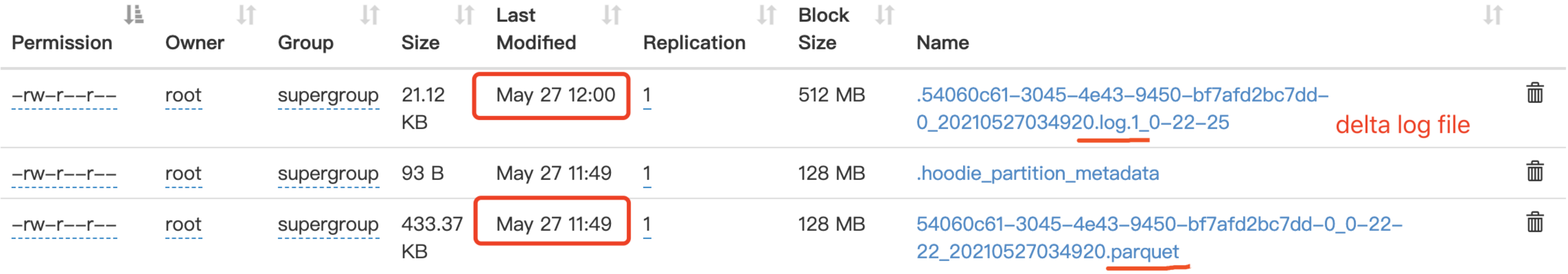
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	93 B	May 27 11:48	1	128 MB	.hoodie_partition_metadata
-rw-r--r--	root	supergroup	433.38 KB	May 27 11:48	1	128 MB	8a97ba16-2bfb-4b16-b815-e5245a2879c1-0_0-22-22_20210527034852.parquet
-rw-r--r--	root	supergroup	433.08 KB	May 27 12:00	1	128 MB	8a97ba16-2bfb-4b16-b815-e5245a2879c1-0_0-22-25_20210527040011.parquet

Version1

Version2

Showing 1 to 3 of 3 entries

COW Table



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	21.12 KB	May 27 12:00	1	512 MB	.54060c61-3045-4e43-9450-bf7afd2bc7dd-0_20210527034920.log.1_0-22-25
-rw-r--r--	root	supergroup	93 B	May 27 11:49	1	128 MB	.hoodie_partition_metadata
-rw-r--r--	root	supergroup	433.37 KB	May 27 11:49	1	128 MB	54060c61-3045-4e43-9450-bf7afd2bc7dd-0_0-22-22_20210527034920.parquet

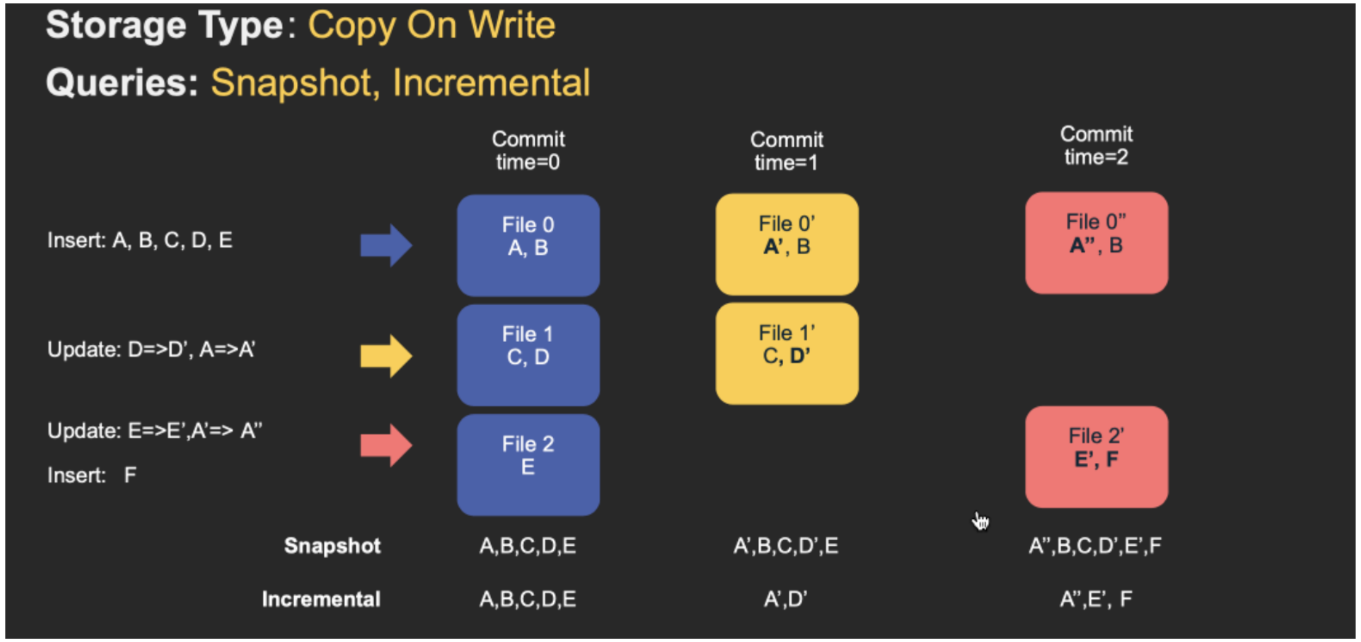
delta log file

MOR Table

https://openlookeng.io

Hudi Tables and Queries

- Query Types
  - **Snapshot Queries**: Queries see the **latest snapshot** of the table as of a given commit or compaction action.
  - **Incremental Queries**: Queries only see **new data** written to the table since a given commit/compaction.
  - **Read Optimized Queries**: Queries see the latest snapshot of a table as of **a given commit/compaction** action.



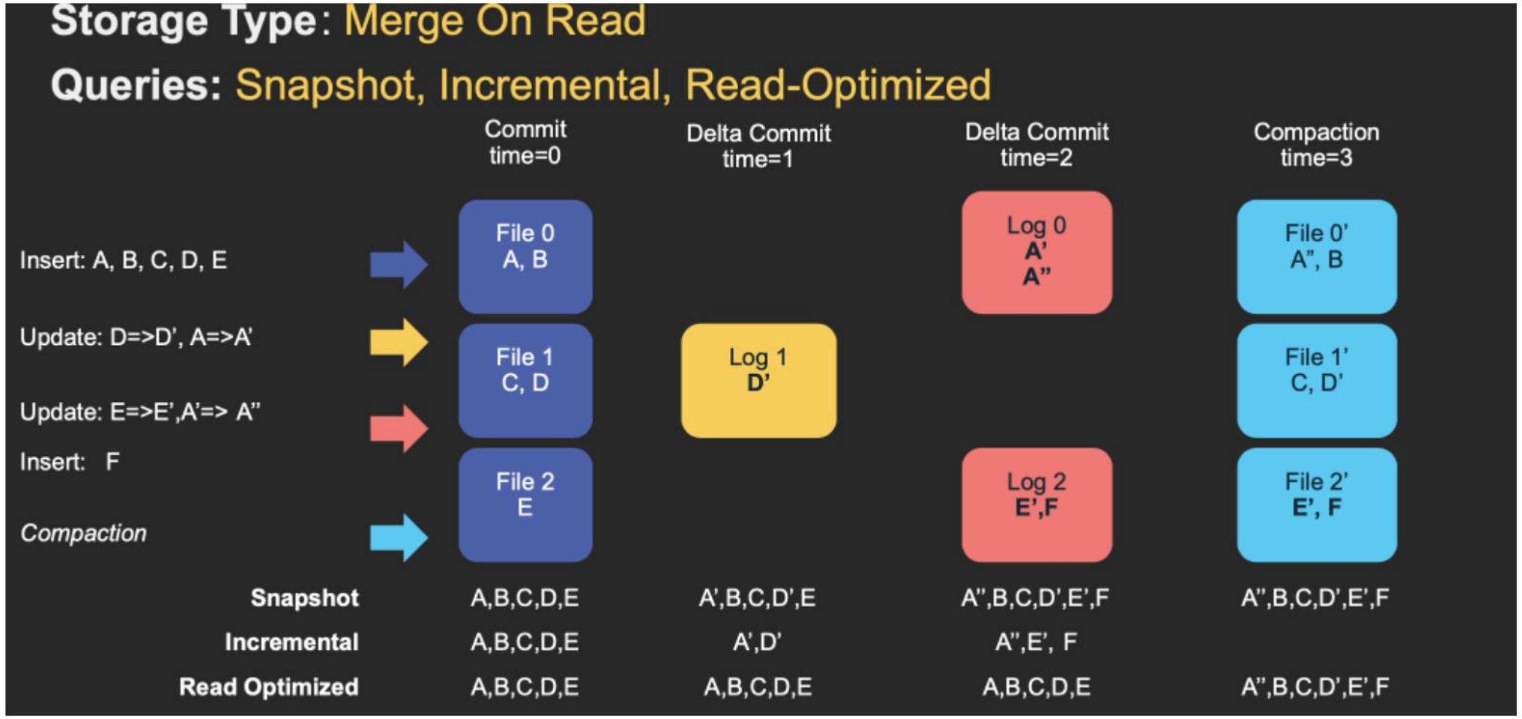
Storage Type: Copy On Write

Queries: Snapshot, Incremental

	Commit time=0	Commit time=1	Commit time=2
Insert: A, B, C, D, E	File 0 A, B	File 0' **A'**, B	File 0" **A"**, B
Update: D=>D', A=>A'	File 1 C, D	File 1' C, **D'**	
Update: E=>E',A'=> A" Insert: F	File 2 E		File 2' **E', F**
Snapshot	A,B,C,D,E	A',B,C,D',E	A",B,C,D',E',F
Incremental	A,B,C,D,E	A',D'	A",E', F

Hudi Tables and Queries

- Query Types
  - **Snapshot Queries**: Queries see the **latest snapshot** of the table as of a given commit or compaction action.
  - **Incremental Queries**: Queries only see **new data** written to the table since a given commit/compaction.
  - **Read Optimized Queries**: Queries see the latest snapshot of a table as of **a given commit/compaction** action.



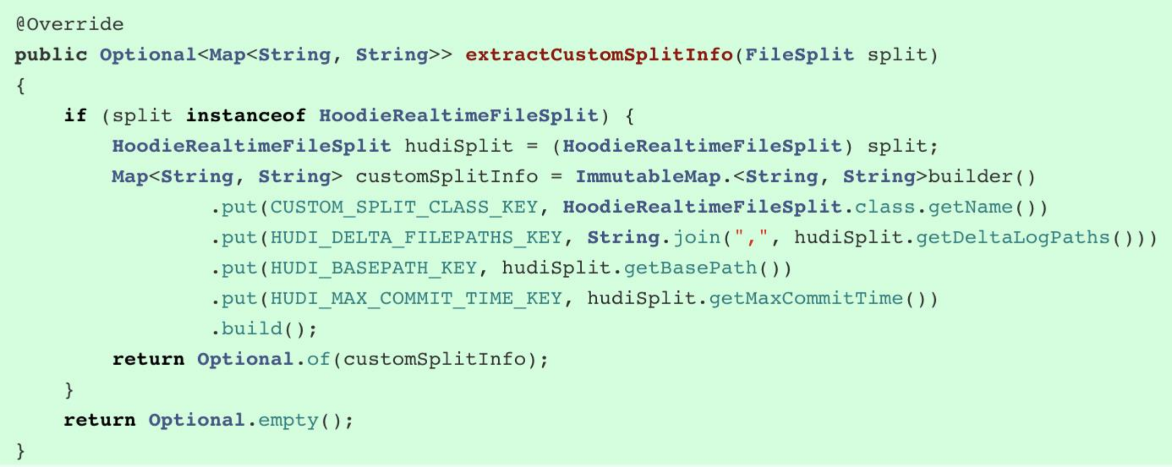
Storage Type: Merge On Read

Queries: Snapshot, Incremental, Read-Optimized

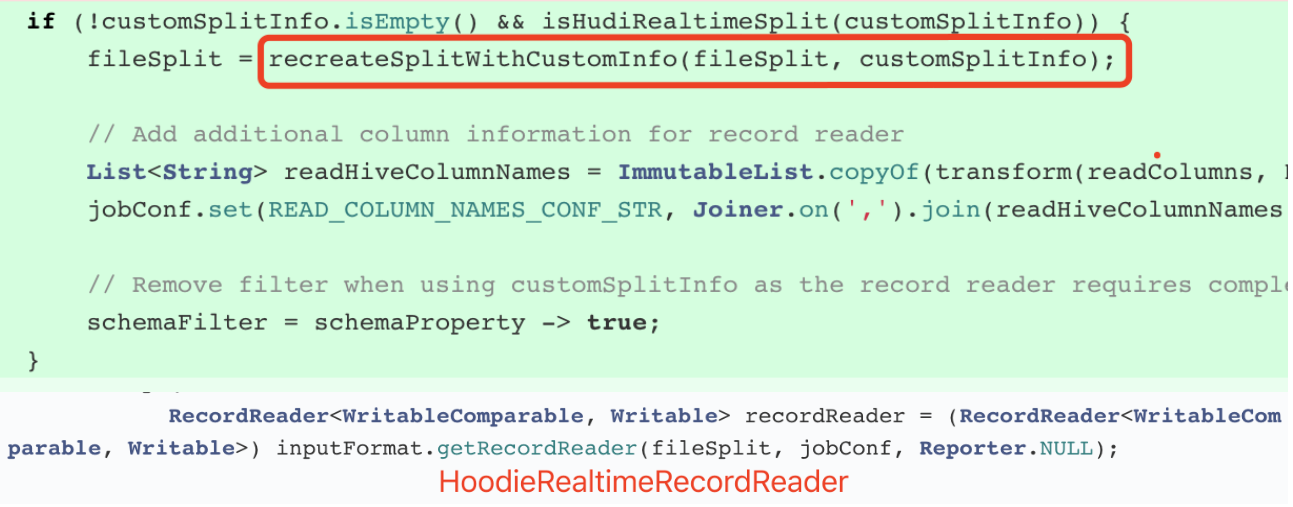
	Commit time=0	Delta Commit time=1	Delta Commit time=2	Compaction time=3
Snapshot	A,B,C,D,E	A’,B,C,D’,E	A”,B,C,D’,E’,F	A”,B,C,D’,E’,F
Incremental	A,B,C,D,E	A’,D’	A”,E’, F	
Read Optimized	A,B,C,D,E	A,B,C,D,E	A,B,C,D,E	A”,B,C,D’,E’,F

Hudi Connector Support

- https://gitee.com/openlookeng/hetu-core/pulls/881
- OpenLooKeng support for querying COW/MOR tables in Hudi
  - possible for querying MOR tables by merging base file (Parquet data) and log files (Avro data) at read time.
- Necessary changes
  - Add extra metadata field to “serializable *HiveSplit*” to store Hudi split information.
  - Recreate Hudi split from *HiveSplit's* extra metadata.
  - Use *HoodieRealtimeRecordReader* to read recreated *HoodieRealtimeFileSplit*



```java
@Override
public Optional<Map<String, String>> extractCustomSplitInfo(FileSplit split)
{
    if (split instanceof HoodieRealtimeFileSplit) {
        HoodieRealtimeFileSplit hudiSplit = (HoodieRealtimeFileSplit) split;
        Map<String, String> customSplitInfo = ImmutableMap.<String, String>builder()
                .put(CUSTOM_SPLIT_CLASS_KEY, HoodieRealtimeFileSplit.class.getName())
                .put(HUDI_DELTA_FILEPATHS_KEY, String.join(",", hudiSplit.getDeltaLogPaths()))
                .put(HUDI_BASEPATH_KEY, hudiSplit.getBasePath())
                .put(HUDI_MAX_COMMIT_TIME_KEY, hudiSplit.getMaxCommitTime())
                .build();
        return Optional.of(customSplitInfo);
    }
    return Optional.empty();
}
```



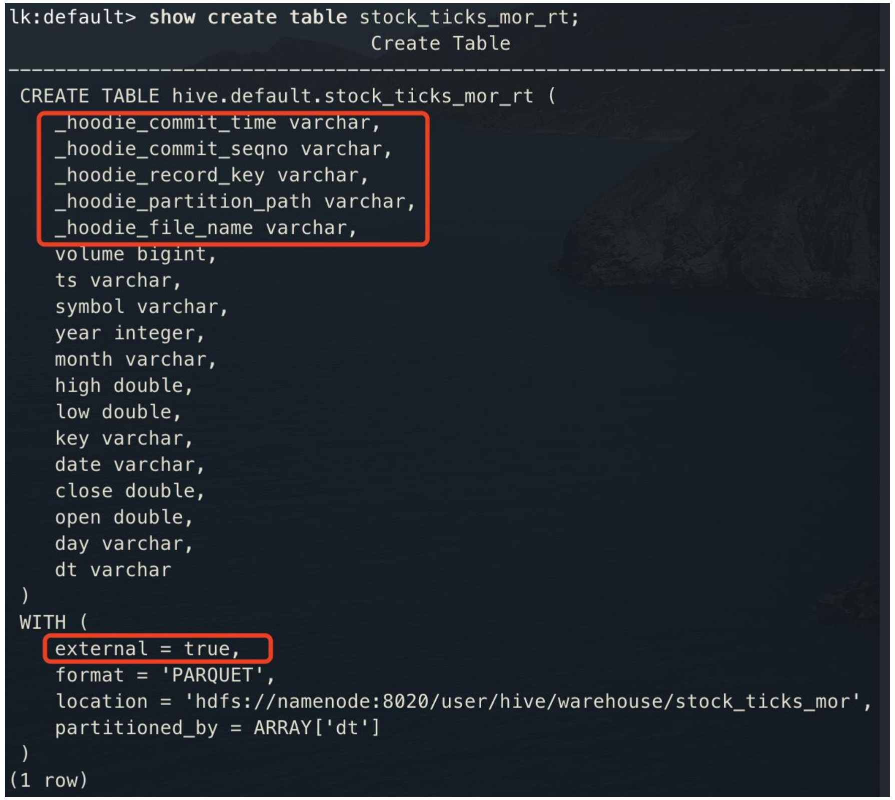
```java
if (!customSplitInfo.isEmpty() && isHudiRealtimeSplit(customSplitInfo)) {
    fileSplit = recreateSplitWithCustomInfo(fileSplit, customSplitInfo);

    // Add additional column information for record reader
    List<String> readHiveColumnNames = ImmutableList.copyOf(transform(readColumns,
    jobConf.set(READ_COLUMN_NAMES_CONF_STR, Joiner.on(',').join(readHiveColumnNames

    // Remove filter when using customSplitInfo as the record reader requires compl
    schemaFilter = schemaProperty -> true;
}
```

```java
            RecordReader<WritableComparable, Writable> recordReader = (RecordReader<WritableCom
parable, Writable>) inputFormat.getRecordReader(fileSplit, jobConf, Reporter.NULL);
```

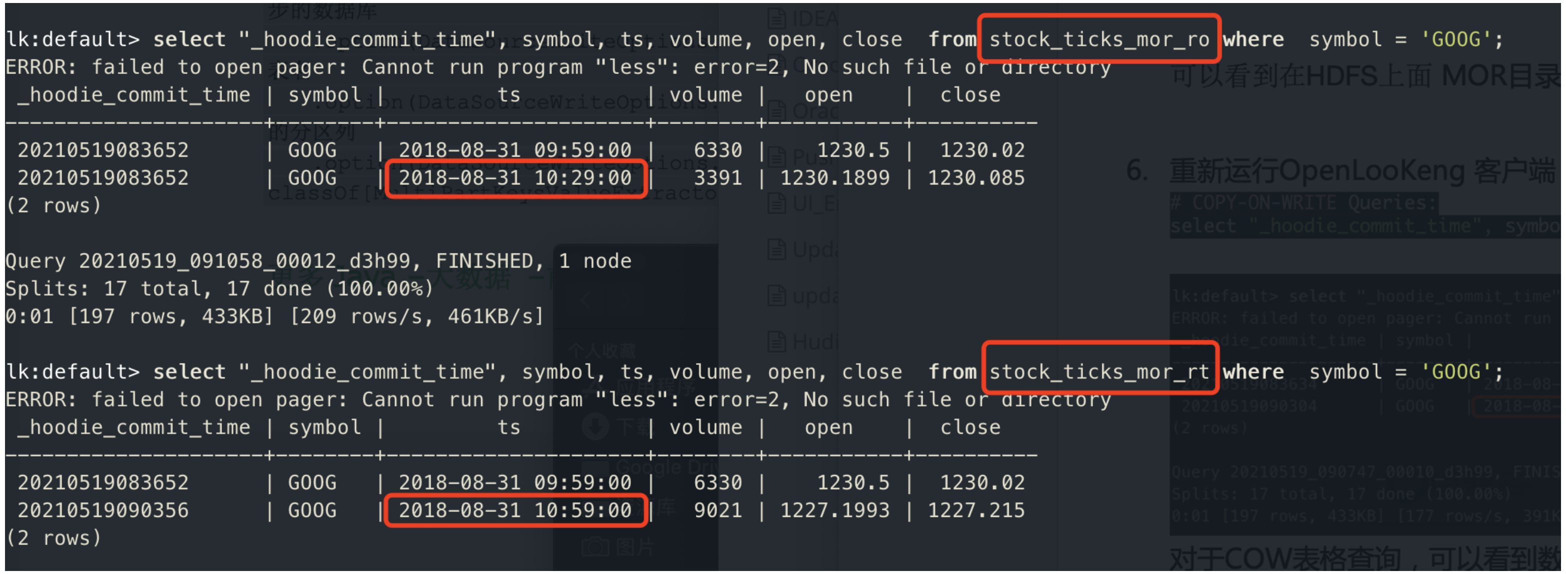
HoodieRealtimeRecordReader

Hudi Connector Support

- https://gitee.com/openlookeng/hetu-core/pulls/881
- Results



```
lk:default> show create table stock_ticks_mor_rt;
                                     Create Table
-----------------------------------------------------------------------------
 CREATE TABLE hive.default.stock_ticks_mor_rt (
    _hoodie_commit_time varchar,
    _hoodie_commit_seqno varchar,
    _hoodie_record_key varchar,
    _hoodie_partition_path varchar,
    _hoodie_file_name varchar,
    volume bigint,
    ts varchar,
    symbol varchar,
    year integer,
    month varchar,
    high double,
    low double,
    key varchar,
    date varchar,
    close double,
    open double,
    day varchar,
    dt varchar
 )
 WITH (
    external = true,
    format = 'PARQUET',
    location = 'hdfs://namenode:8020/user/hive/warehouse/stock_ticks_mor',
    partitioned_by = ARRAY['dt']
 )
(1 row)
```



```
lk:default> select "_hoodie_commit_time", symbol, ts, volume, open, close  from stock_ticks_mor_ro where  symbol = 'GOOG';
ERROR: failed to open pager: Cannot run program "less": error=2, No such file or directory
 _hoodie_commit_time | symbol |         ts          | volume |   open    |  close
---------------------+--------+---------------------+--------+-----------+----------
 20210519083652      | GOOG   | 2018-08-31 09:59:00 |   6330 |    1230.5 |  1230.02
 20210519083652      | GOOG   | 2018-08-31 10:29:00 |   3391 | 1230.1899 | 1230.085
(2 rows)

Query 20210519_091058_00012_d3h99, FINISHED, 1 node
Splits: 17 total, 17 done (100.00%)
0:01 [197 rows, 433KB] [209 rows/s, 461KB/s]

lk:default> select "_hoodie_commit_time", symbol, ts, volume, open, close  from stock_ticks_mor_rt where  symbol = 'GOOG';
ERROR: failed to open pager: Cannot run program "less": error=2, No such file or directory
 _hoodie_commit_time | symbol |         ts          | volume |   open    |  close
---------------------+--------+---------------------+--------+-----------+----------
 20210519083652      | GOOG   | 2018-08-31 09:59:00 |   6330 |    1230.5 |  1230.02
 20210519090356      | GOOG   | 2018-08-31 10:59:00 |   9021 | 1227.1993 | 1227.215
(2 rows)
```

https://openlookeng.io

Live Demo

openLooKeng

https://openlookeng.io

Reference

- 1. https://eng.uber.com/hoodie/
- 2. https://prestodb.io/blog/2020/08/04/prestodb-and-hudi
- 3. https://www.cnblogs.com/leesf456/p/13710005.html
- 4. https://hudi.apache.org/docs/docker_demo.html
- 5. https://gitee.com/openlookeng/hetu-core/pulls/881

Thank you!

openLooKeng

https://openlookeng.io