Hosted by Starburst

Trino Fest

Enhancing Trino's Query Performance and Data Management with Hudi: Innovations and Future

June 13, 2024

Ethan Guo

ethan@onehouse.ai

Speaker Bio

Ethan Guo

- ➔ Data Infrastructure Engineer @ Onehouse.ai
- ➔ Apache Hudi PMC Member
- ➔ Senior Engineer @ Uber

  Data (Near Real-Time Analytics with Hudi Incremental Processing),

  Networking (App Network Performance with QUIC)

in/yihua-ethan-guo/

Trino + Hudi: Fast Analytics + Upserts

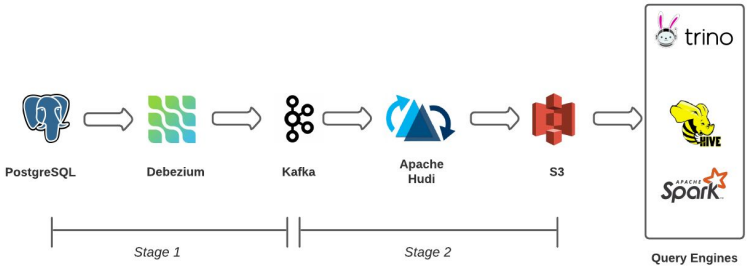
Trino

- Fast SQL query with massively parallel processing

Hudi

- Fast upserts with incremental processing in Lakehouse

Fresher data, reports, and analytics



Robinhood's architecture and use cases for Trino and Hudi: https://trino.io/episodes/41.html

Agenda

- Apache Hudi: The Open Data Lakehouse Platform
- Improving Query Performance with Multi-Modal Index in Hudi
- Enhancing Trino Hudi Connector
- Future of Trino with Hudi

Apache Hudi:
The Open Data Lakehouse Platform

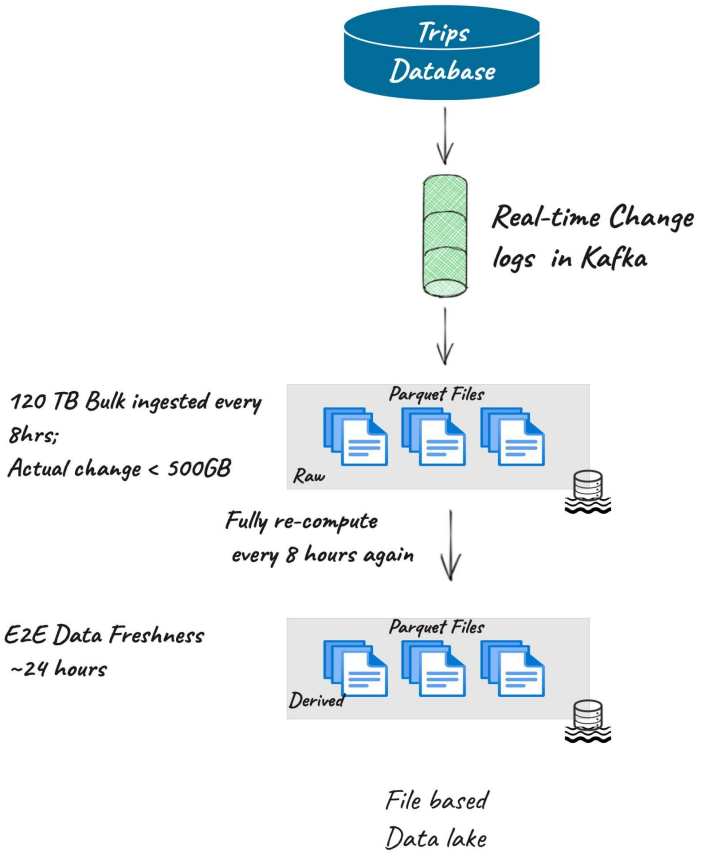
Origins@Uber 2016

Context

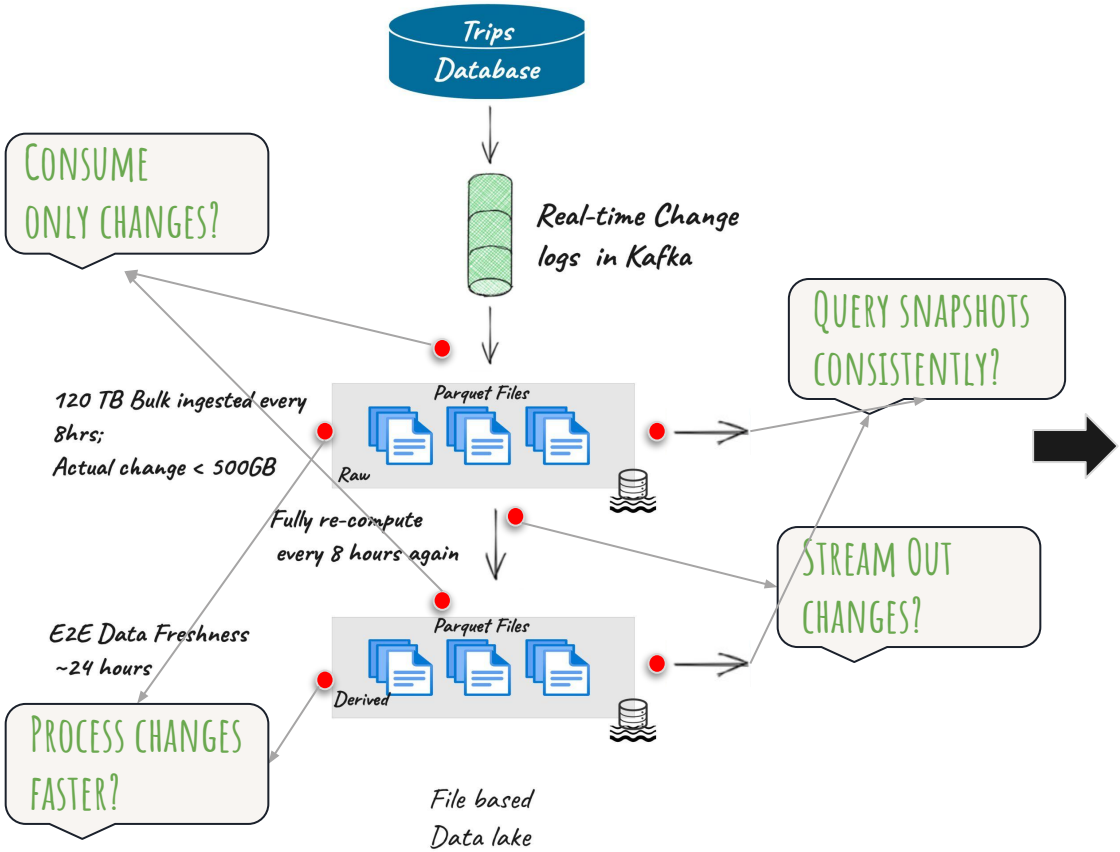
- Uber in hypergrowth
- Moving from warehouse to lake
- HDFS/Cloud storage is immutable

Problems

- Extremely poor ingest performance
- Wasteful reading/writing
- Zero concurrency control or ACID



Missing pieces: Upserts, Deletes & Incrementals



Core Primitives in Hudi

- **Upserts:** Absorb changes to records and process faster
- **Incremental Reads:** Obtain records that changed
- **Snapshot isolation:** Read latest committed state consistently

Apache hudi The Lakehouse Platform

Apache hudi Proven @ Massive Scale

ByteDance

https://chowdera.com/2022/184/202207030146453436.html
https://hudi.apache.org/blog/2021/09/01/building-eb-level-data-lake-using-hudi-at-bytedance/

100GB/s
Throughput

> 1Exabyte
Even just 1 Table

70%
CPU Savings
(write+read)

Daily -> Min
Analytics Latency

Walmart

https://www.youtube.com/watch?v=ZamXiT9aqs8

300GB/d
Throughput

25+TB
Datasets

Hourly
Analytics Latency

Uber

https://www.uber.com/blog/apache-hudi-graduation/

4000+
Tables

250+PB
Raw + Derived

800B
Records/Day

Daily -> Min
Analytics Latency

GE Aviation

https://aws.amazon.com/blogs/big-data/how-ge-aviation-built-cloud-native-data-pipelines-at-enterprise-scale-using-the-aws-platform/

10,000+
Tables

150+
Source systems

CDC, ETL
Use cases

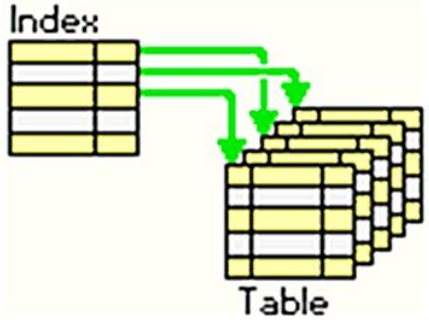
Improving Query Performance with Multi-Modal Index in Hudi

Improving Query Performance

Key: Reading fewer bytes from Input Tables

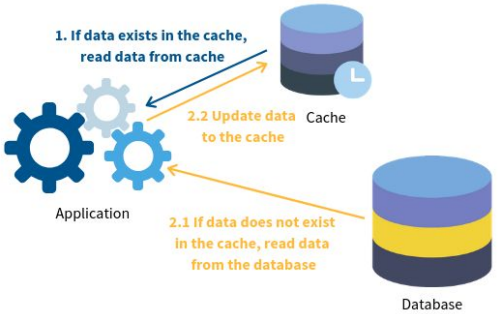
Indexes

- Helpful for selective queries i.e needles in haystacks
- B-trees, bloom-filters, bit-maps..



Caching

- Eliminate access to storage in the common case
- Read-through, write-through, columnar vs row based

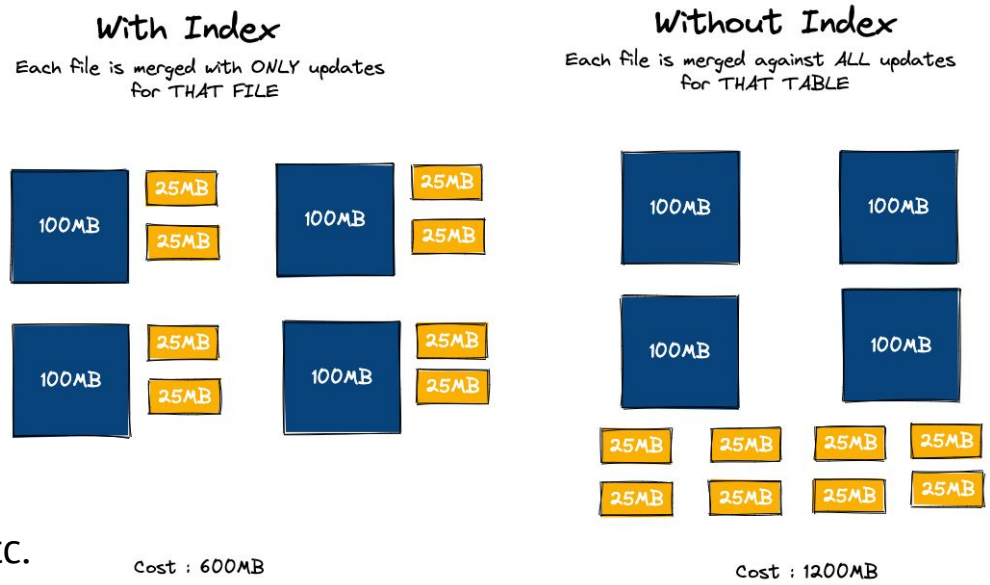


Storage Layout

- Control how data is physically organized in storage
- Bucketing, Clustering

Indexes: Locating Records Efficiently

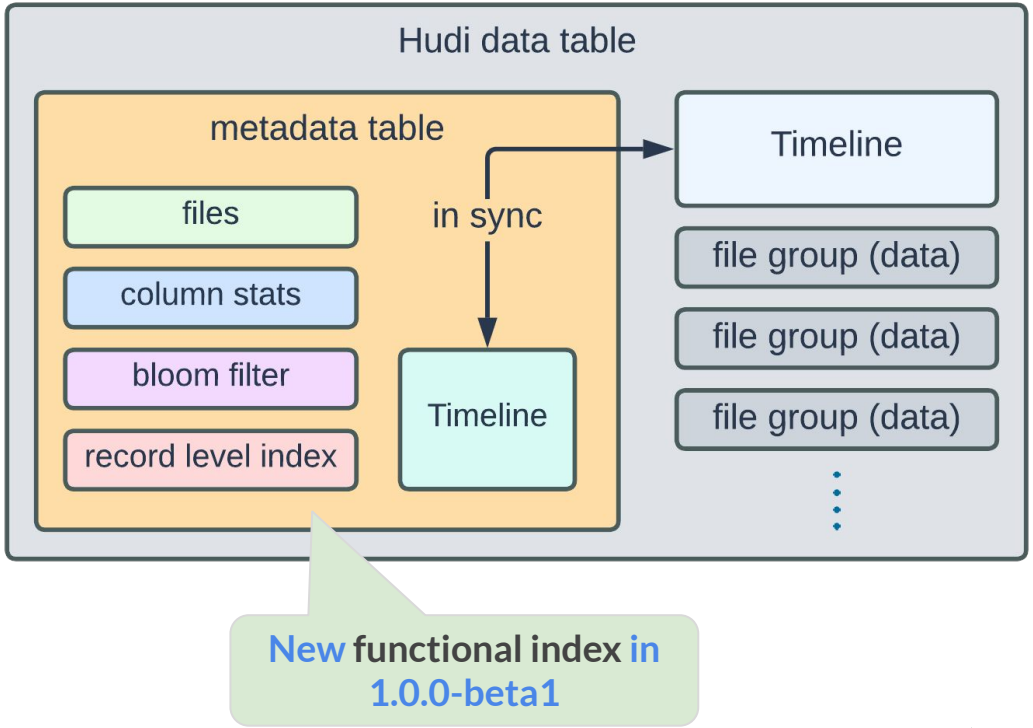
- Widely employed in DB systems
  - Locate information quickly
  - Reduce I/O cost
  - Improve Query efficiency
- Indexing provides fast upserts
  - Locate records for incoming writes
  - Bloom filter based, Simple, Hbase, etc.



https://hudi.apache.org/blog/2020/11/11/hudi-indexing-mechanisms/

Multi-Modal Index with Metadata Table

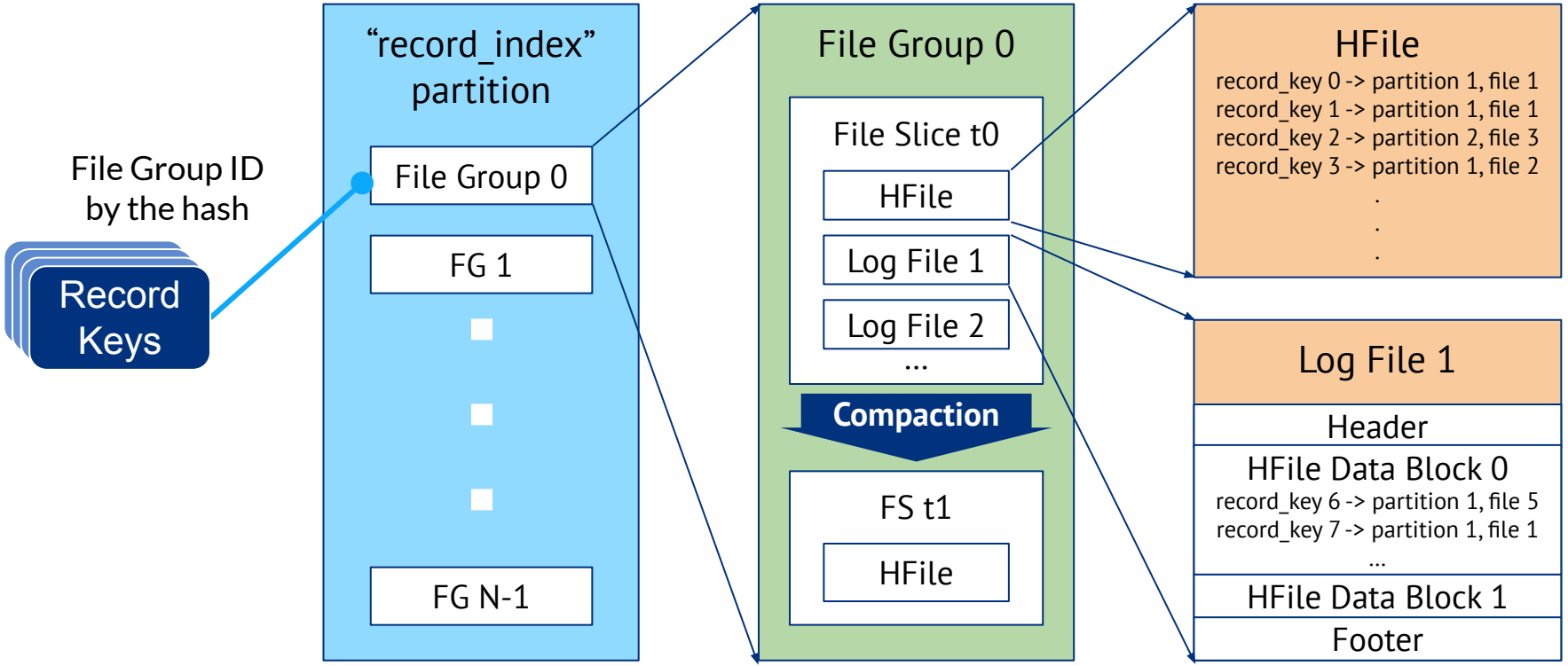
- Partitioned for extensibility
  - Files
  - Column stats
  - Bloom filter
  - Record index
  - Functional index
- Support CREATE/DROP index
- Support async indexing



Record-Level Index (RLI) - New in Hudi 0.14

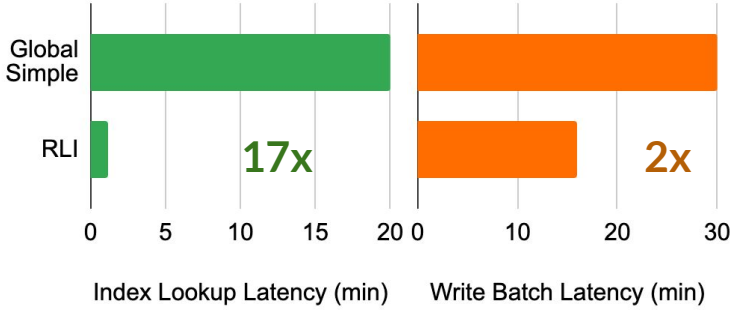
- Challenges
  - Reading data and metadata per file is expensive
  - HBase index requires cluster maintenance which is operationally difficult
- Design
  - Key-to-location mapping in table-level metadata
    - A new partition, “record_index”, in the metadata table
    - Stored in a few file groups instead of all data files
  - Fast index update and lookup
    - MDT, an internal Hudi MOR table, enables uniformed fast updates
    - HFile format enables fast point lookup

Record-Level Index on Storage



Performance Benefit from RLI

- Improves index lookup and write latency
  - 1TB dataset, 200MB batch, random updates, Spark datasource
  - **17x** speedup on index lookup, **2x** on write
- Reduces SQL latency with point lookups
  - TPC-DS 10TB datasets, store_sales table, Spark
  - **2-3x** improvement compared to no RLI



SELECT * FROM table WHERE key = 'val'
DELETE FROM table WHERE key = 'val'

[RLI blog: Hudi's blazing fast indexing for large-scale datasets]

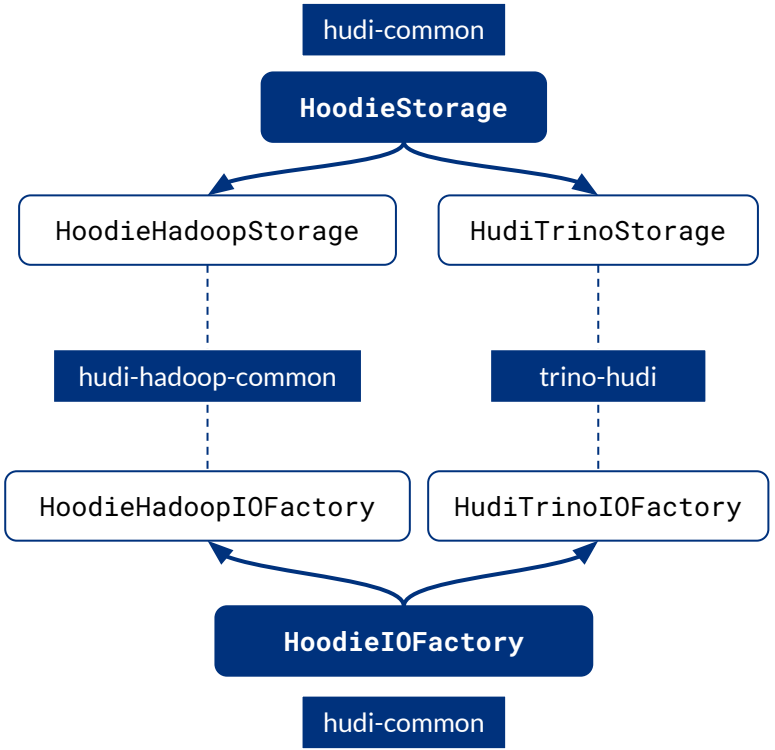
Enhancing Trino Hudi Connector

Hudi Support in Trino

- Hive connector
  - Hudi integration through InputFormat implementation
  - COW, MOR read-optimized, snapshot, and bootstrap queries (deprecated in v411, redirects to Hudi connector)
- Hudi connector
  - COW, MOR read-optimized queries only since v398; no support of metadata-based (MDT) file listing since v419
  - Due to removal of Hudi dependencies as part of Trino dehadooping
  - **RO, snapshot, bootstrap query support with MDT in upcoming Trino releases**

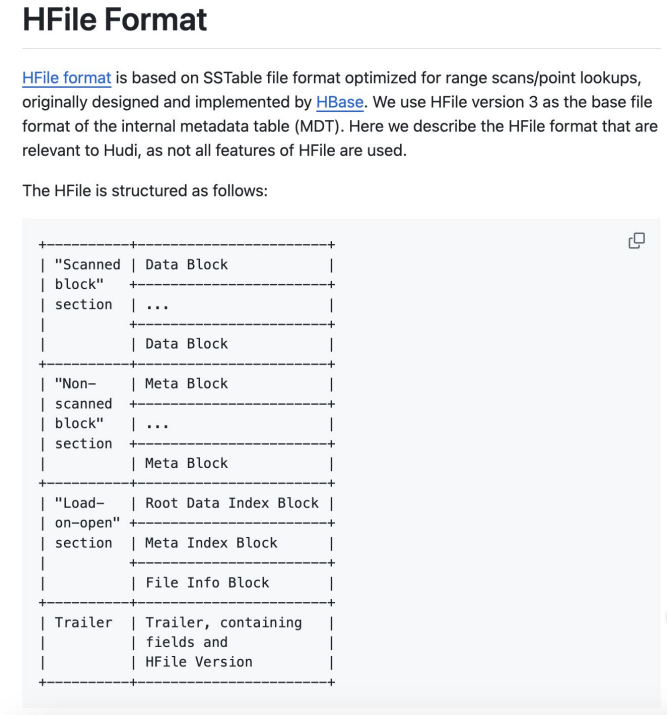
Hudi Storage Abstraction - New in Hudi 0.15

- `HoodieStorage` abstraction
  - Hadoop-independent file system and storage APIs
  - Extendable with Hadoop FileSystem and TrinoFileSystem
- `HoodieIOFactory` abstraction
  - Creates readers and writers for I/O (e.g., HFile)
- Hadoop-independent hudi-common module for reader integration
  - Plugs in storage and factory implementations



New HFile Reader - New in Hudi 0.15

- HFile Format Spec
  - Defines the HFile Format required by Hudi to enable fast point lookups in MDT
  - Custom HFile implementation (e.g., in C++ or Rust) possible by following the Spec
- New HFile Reader implementation in Java
  - Independent of HBase or Hadoop dependencies
  - Backwards compatible with existing Hudi releases and storage format



HFile Format

HFile format is based on SSTable file format optimized for range scans/point lookups, originally designed and implemented by HBase. We use HFile version 3 as the base file format of the internal metadata table (MDT). Here we describe the HFile format that are relevant to Hudi, as not all features of HFile are used.

The HFile is structured as follows:

```
+----------+------------------------+
| "Scanned | Data Block             |
| block"   +------------------------+
| section  | ...                    |
|          +------------------------+
|          | Data Block             |
+----------+------------------------+
| "Non-    | Meta Block             |
| scanned  +------------------------+
| block"   | ...                    |
| section  +------------------------+
|          | Meta Block             |
+----------+------------------------+
| "Load-   | Root Data Index Block  |
| on-open" +------------------------+
| section  | Meta Index Block       |
|          +------------------------+
|          | File Info Block        |
+----------+------------------------+
| Trailer  | Trailer, containing    |
|          | fields and             |
|          | HFile Version          |
+----------+------------------------+
```

HFile Format Spec in Hudi:
https://github.com/apache/hudi/blob/master/hudi-io/hfile_format.md

Trino Hudi Connector Integration

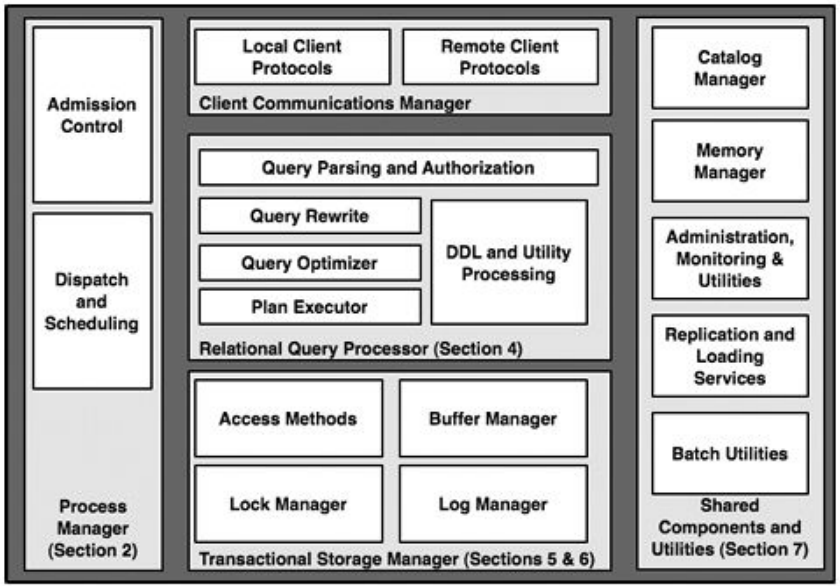
- Re-introduce hudi-common dependency
  - Makes Hudi support maintainable
  - Evolves easily with future storage format changes
  - Hadoop-independent with `TrinoFileSystem`, unlocks optimization like caching
- Support MDT-based file listing
  - Uses new HFile Reader to support MDT read and lookup
  - **38% query latency reduction*** on Trino Hudi connector in TPC-DS 1TB benchmark
- Support MOR snapshot query
  - `HudiDirectoryLister` determines the file listing
  - New `HudiSnapshotDirectoryLister` implementation for snapshot queries

* based on Trino Hudi Connector feature branch; we'll upstream the changes.

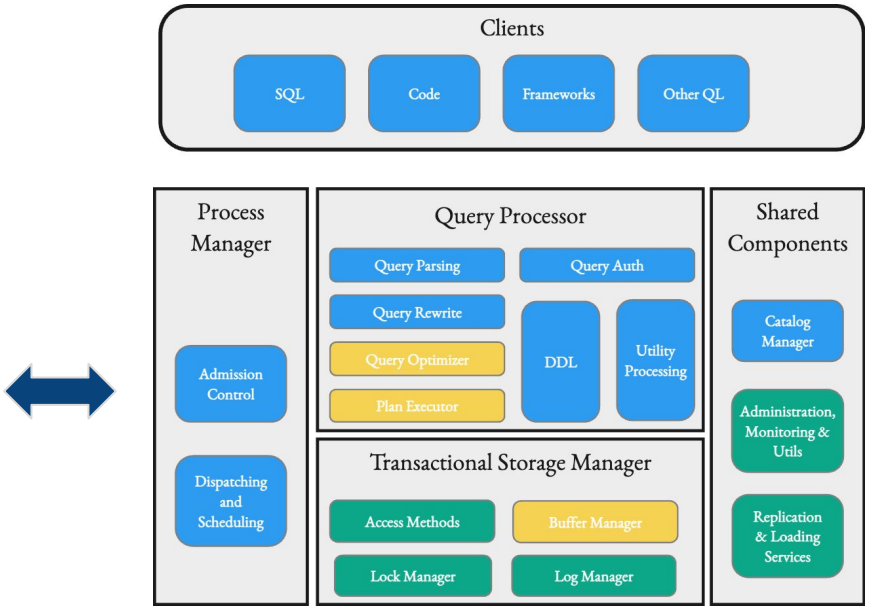
Future of Trino with Hudi

Hudi 1.x - Database for the Lakehouse

"Reimagination of Hudi, as the *transactional database for the lake*, with polyglot persistence"



Main components of a DBMS.
***Courtesy**: The seminal database paper: Architecture of a Database System*



Reference diagram highlighting existing (green) and new (yellow) Hudi components, along with external components (blue). Checkout RFC-69

New Indexes in Hudi 1.x

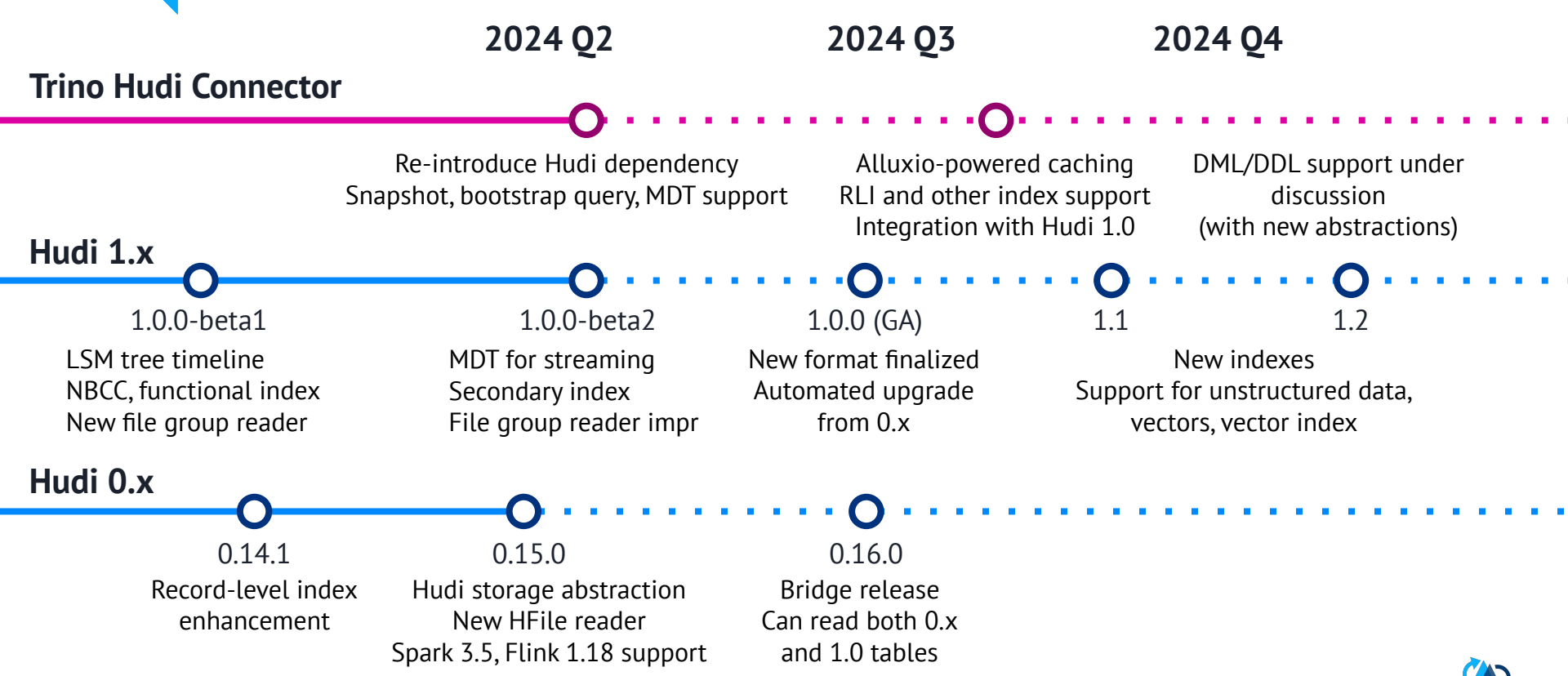
- Functional index (RFC-63, in 1.0.0-beta1)
  - Relational databases allow index on functions or expressions
  - Accelerate queries based on results of computations
  - Absorb partitioning into indexes
  - No more hide-and-evolving partitions!
- Secondary index (RFC-77, in 1.0.0-beta2)
  - Index for non-key fields
  - Improves query performance with predicates on the fields with secondary index built

```
CREATE INDEX datestr ON hudi_table USING
column_stats(ts) options(func='from_unixtime',
format='yyyy-MM-dd');
```

Physical partition path	File Name	Min of datestr	Max of datestr	Note
org_id=1/datestr=2022-10-01/	base_file_1.parquet	2022-10-01	2022-10-01	Old partitioning scheme
org_id=1/datestr=2022-10-02/	base_file_2.parquet	2022-10-02	2022-10-02	
org_id=2/datestr=2022-10-01/	base_file_3.parquet	2022-10-01	2022-10-01	
org_id=3/datestr=2022-10-01/	base_file_4.parquet	2022-10-01	2022-10-01	
...	...	...	...	...
org_id=1/	base_file_10.parquet	2022-10-10	2022-10-11	New partitioning scheme
org_id=2/	base_file_11.parquet	2022-10-10	2022-10-15	
...	...	...	...	...

```
CREATE INDEX idx_city ON hudi_table USING
secondary_index(city);
```

Roadmap


2024 Q2
2024 Q3
2024 Q4
Trino Hudi Connector
Re-introduce Hudi dependency
Snapshot, bootstrap query, MDT support
Alluxio-powered caching
RLI and other index support
Integration with Hudi 1.0
DML/DDL support under discussion
(with new abstractions)
Hudi 1.x
1.0.0-beta1
LSM tree timeline
NBCC, functional index
New file group reader
1.0.0-beta2
MDT for streaming
Secondary index
File group reader impr
1.0.0 (GA)
New format finalized
Automated upgrade from 0.x
1.1
New indexes
Support for unstructured data, vectors, vector index
1.2
Hudi 0.x
0.14.1
Record-level index enhancement
0.15.0
Hudi storage abstraction
New HFile reader
Spark 3.5, Flink 1.18 support
0.16.0
Bridge release
Can read both 0.x and 1.0 tables

Come Build With The Community!

Docs : https://hudi.apache.org

Blogs : https://hudi.apache.org/blog

Slack : Apache Hudi Slack Group

Twitter : https://twitter.com/apachehudi

Github: https://github.com/apache/hudi/ Give us a star ⭐!

Mailing list(s) :

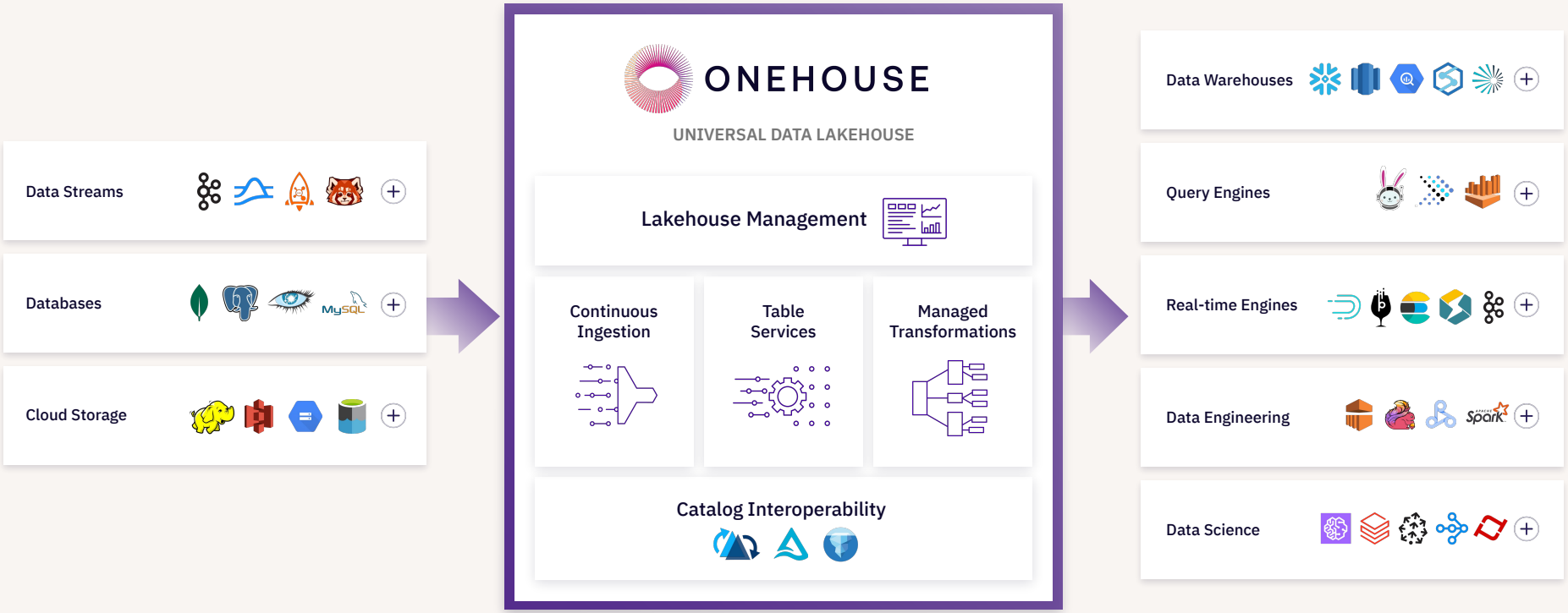
dev-subscribe@hudi.apache.org (send an empty email to subscribe)

Join Hudi Slack

The Onehouse Universal Data Lakehouse

Delivered as a Fully-Managed Cloud Service

Swing by Onehouse booth at Trino Fest 2024



Hosted by Starburst

Trino Fest

Enhancing Trino's Query Performance and Data Management with Hudi: Innovations and Future

Join Hudi Slack

Thanks!

Questions?