
Apache
hudi
VS
DELTA LAKE
ICEBERG

Origin Stories

2017 — open sourced — Uber

```
+ # Hudi
+ Hudi (pronounced Hoodie) stands for `Hadoop Upserts anD Incrementals`. Hudi manages storage of large analytical datasets on [HDFS](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html) and serve them out via two types of tables

  * **Read Optimized Table** - Provides excellent query performance via purely columnar storage (e.g. [Parquet](https://parquet.apache.org/))
  * **Near-Real time Table (WIP)** - Provides queries on real-time data, using a combination of columnar & row based storage (e.g Parquet + [Avro](http://avro.apache.org/docs/current/mr.html))
```

2018 — open sourced — N

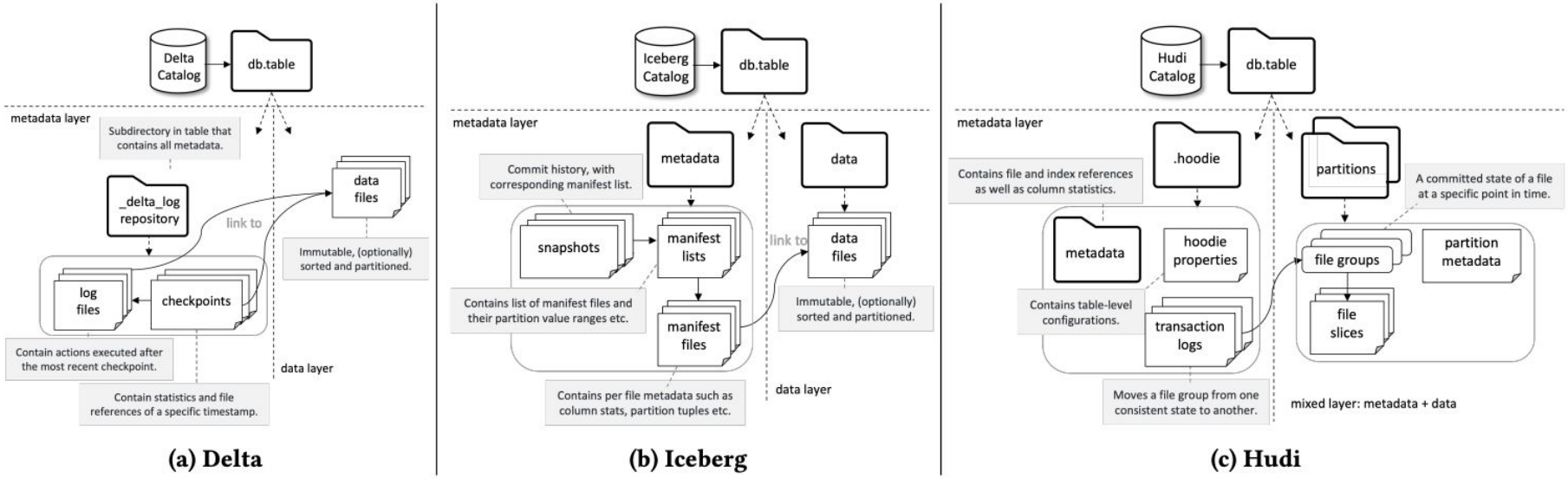
```
+ ## Iceberg
+
+ Iceberg is a new table format for storing large, slow-moving tabular data. It is designed to improve on the de-facto standard table layout built into Hive, Presto, and Spark.
+
```

2019 — open sourced

```
- Delta Lake Core is .... (copy text from delta docs)
+ Delta Lake is a next-generation engine built on top of Apache Spark. Delta Lake provides ACID transactions, optimized layouts and indexes, and execution engine improvements for building data pipelines to support big data use cases: batch and streaming ingests, fast interactive queries, and machine learning. Specifically, Delta offers:
```

Technical Fundamentals

- Metadata abstractions on files in cloud object storage
- Tables with SQL semantics and schema evolution
- ACID transactions
- Updates and deletes (merge/upsert)
- Data layout optimizations for performance tuning



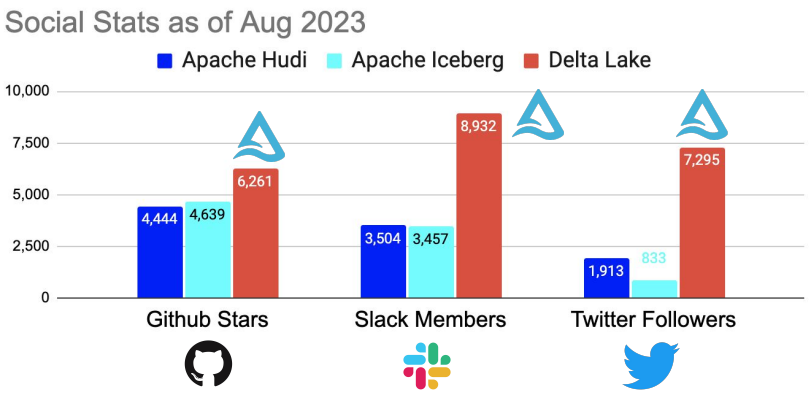
Feature Comparisons

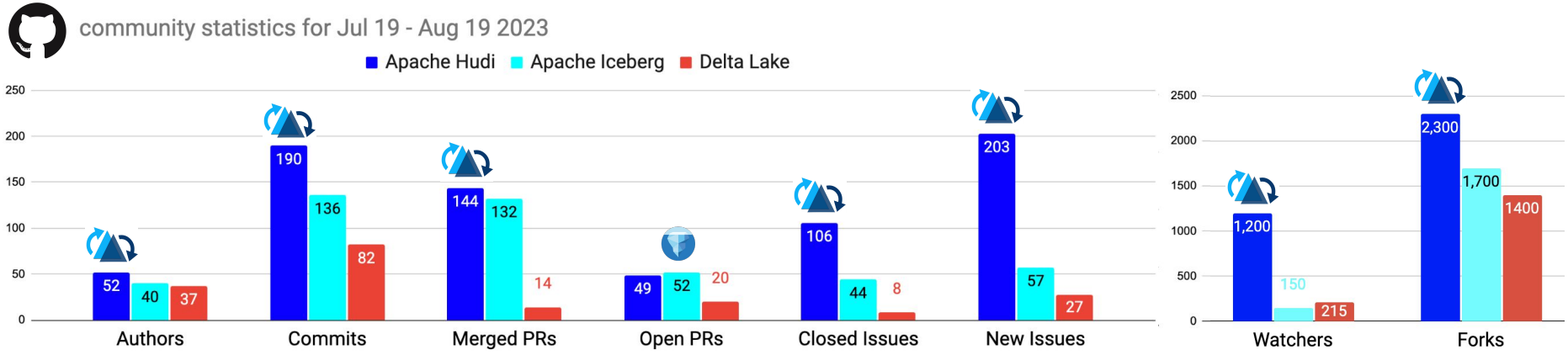
https://www.onehouse.ai/blog/apache-hudi-vs-delta-lake-vs-apache-iceberg-lakehouse-feature-comparison

	Apache hudi As of v0.12.2	DELTA LAKE As of v2.2.0	ICEBERG As of v1.1.0
Read/write features			
ACID Transactions	✓	✓	✓
Copy-On-Write	✓ Writes	✓ Writes	✓ Writes
Merge-On-Read	✓ Merge-On-Read	✗	⚠ Limited functionality
Efficient Bulk Load	✓ Bulk_Insert	✗	✗
Efficient merge with indices	✓ Over 4 types of indexing	✗ Bloom filter index still proprietary	✗ Metadata indexing is for
Bootstrap (Can I upgrade data in-place into the system without rewriting the data?)	✓ Bootstrap	✓ Convert to delta	✓ Table migration
Incremental Query (can I obtain a change stream for a given time window on the table?)	✓ Incremental Query	⚠ CDF Experimental mode after 2.0.0	✗ Can only incrementally read appends

Community Statistics

- Delta Lake has the most public attention
- Apache Hudi has most community engagement and contribution





** Hudi uses Github+JIRA for issues*

Benchmark or Benchmark-eting?

Benchmark Rule #1:

Always test your own workload

Largest benchmark gap:

Most use append-only workload for TPC-DS

Most common mistake:

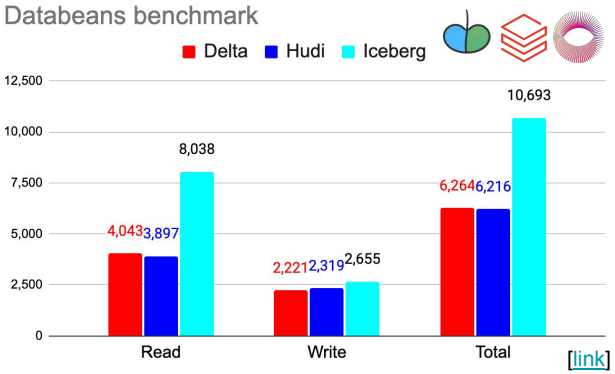
Hudi is by default "upsert"
Comparing upsert vs insert is 🍎 vs 🍊

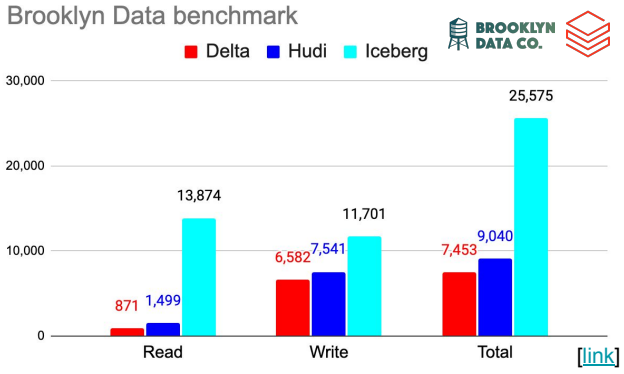
Best benchmark framework:

LST-Bench by Microsoft

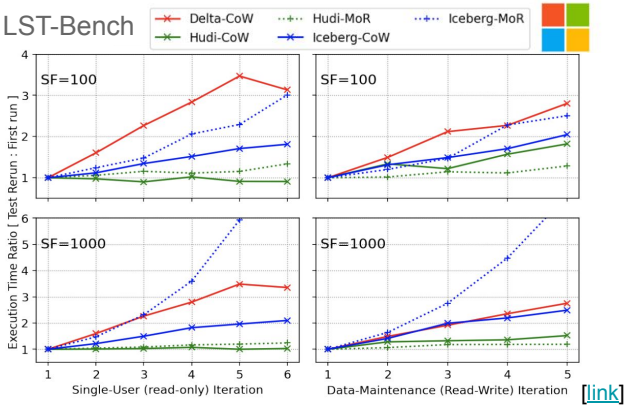
Most common pattern:

Hudi and Delta usually close
Iceberg usually slowest





Measures perf, resiliency, concurrency, and simulates real workloads



Delta failed OCC background compaction…
Iceberg failed writes altogether…

Walmart

	Delta OSS	Apache Iceberg	Apache Hudi
WL-2	DELTA LAKE	ICEBERG	
Query	Due to daily ordering and compaction query performance is optimized.	Due to significant issues with ingestion query testing was not performed	Performace on Count Distinct biased output results, remaining querys ran closer to Delta. However Delta was most performant across all query types
Ingestion	Delta was not able to acomplish dual writes for this workload	Unable to perform cleanup, significant issues in setting up runtime to deliver optimal file sizes	Hudi Provides the best ingestion performance

[link]

Cost Savings Examples

Uber

4/16/23

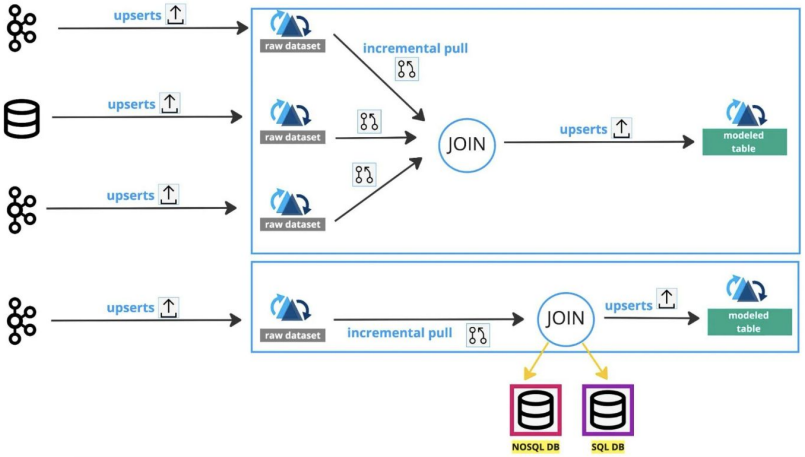
Engineering, Data / ML

Setting Uber's Transactional Data Lake in Motion with Incremental ETL Using Apache Hudi

~80% overall compute cost reductions over millions of v_cores

https://www.uber.com/blog/ubers-lakehouse-architecture/

Incremental Processing



zoom

5/16/23

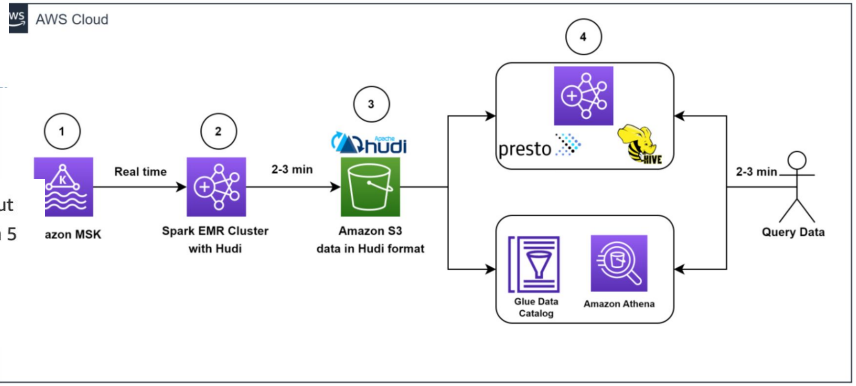
AWS Big Data Blog

How Zoom implemented streaming log ingestion and efficient GDPR deletes using Apache Hudi on Amazon EMR

~80% compute and ~90% storage cost savings

We showed that the costs of EMR clusters can be reduced by about 82% while bringing the storage costs down by about 90% compared to the prior HDFS-based architecture. All of this while making the data available in the data lake within 5 minutes of ingestion from the source. We also demonstrated that data deletions from a data lake containing multiple petabytes of data can be performed much more efficiently. With our optimized approach, we were able to delete approximately 1,000 records in just 1–2 minutes, as compared to the previously required 3 hours or more.

https://aws.amazon.com/blogs/big-data/how-zoom-implemented-streaming-log-ingestion-and-efficient-gdpr-deletes-using-apache-hudi-on-amazon-emr/



ONETABLE

What is Onetable?
Omni-directional interop between Hudi, Delta, Iceberg

Why Onetable?
Initially created for Onehouse customers to enjoy a future proof lakehouse that works with any engine like Databricks and Snowflake

How does it work?
Metadata translation with zero-copying of underlying parquet files

What's next?
Onetable is running in production today for Onehouse customers.

Standalone OSS Github repo releasing soon! We are hardening Onetable with co-owners who are vested in Delta and Iceberg.

If you have exp with hudi, delta, or iceberg and want to be an early contributor, send me a msg

Which format to choose?

Choose Apache Hudi if:

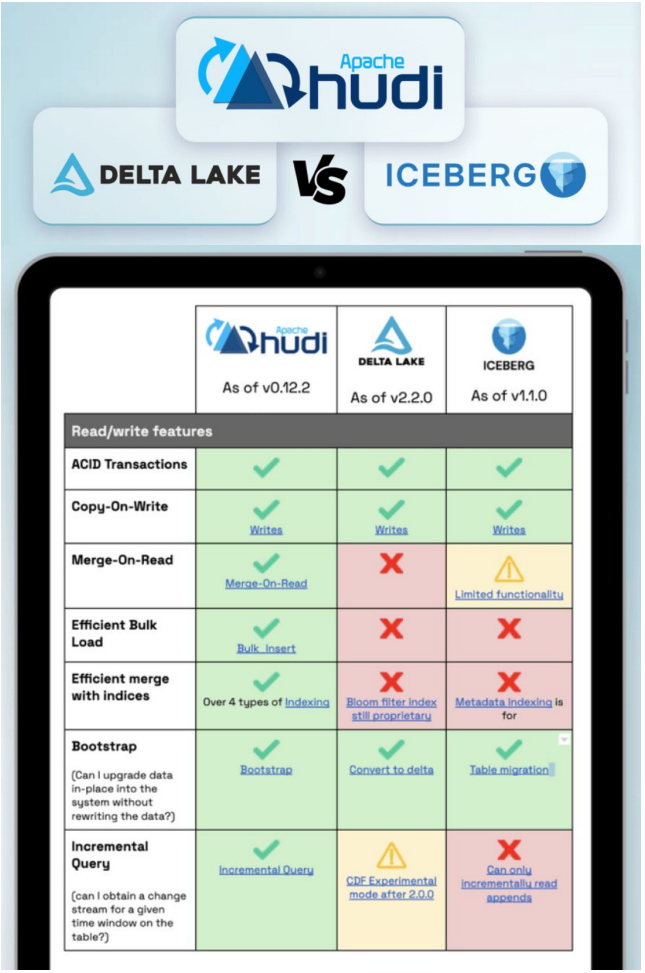
1. Mutable data - GDPR Deletes, Updates
2. CDC workloads
3. Low latency requirements
4. Large ETL pipelines - perf/cost w/ incremental ETL

Choose DELTA LAKE if:

1. Is a current Databricks customer
2. Needs fastest premium Spark with Photon
3. Wants an “easy-to-get-started” table format

Choose ICEBERG if:

1. Trino or Athena writes
2. Snowflake writes
3. Not sensitive to performance
4. Partition evolution



	Apache Hudi As of v0.12.2	DELTA LAKE As of v2.2.0	ICEBERG As of v1.1.0
Read/write features			
ACID Transactions	✓	✓	✓
Copy-On-Write	✓ Writes	✓ Writes	✓ Writes
Merge-On-Read	✓ Merge-On-Read	✗	⚠ Limited functionality
Efficient Bulk Load	✓ Bulk_Insert	✗	✗
Efficient merge with indices	✓ Over 4 types of Indexing	✗ Bloom filter index still proprietary	✗ Metadata indexing is for
Bootstrap (Can I upgrade data in-place into the system without rewriting the data?)	✓ Bootstrap	✓ Convert to delta	✓ Table migration
Incremental Query (can I obtain a change stream for a given time window on the table?)	✓ Incremental Query	⚠ CDF Experimental mode after 2.0.0	✗ Can only incrementally read appends

How to learn more

Build a Hudi, Delta, or Iceberg, lakehouse in minutes: https://onehouse.ai/product

Read More Blogs : https://onehouse.ai/blog/

Chat on Slack : https://join.slack.com/t/apache-hudi/shared_invite/zt-20r833rxh-627NWYDUyR8jRtMa2mZ~gg

Follow us on Twitter : https://twitter.com/onehousehq

Level-Up on LinkedIn: https://www.linkedin.com/company/80166284/

Contact Us: info@onehouse.ai