Decipher Delta Lake

By Nisha Kumari and Divya Dua

ThoughtWorks®

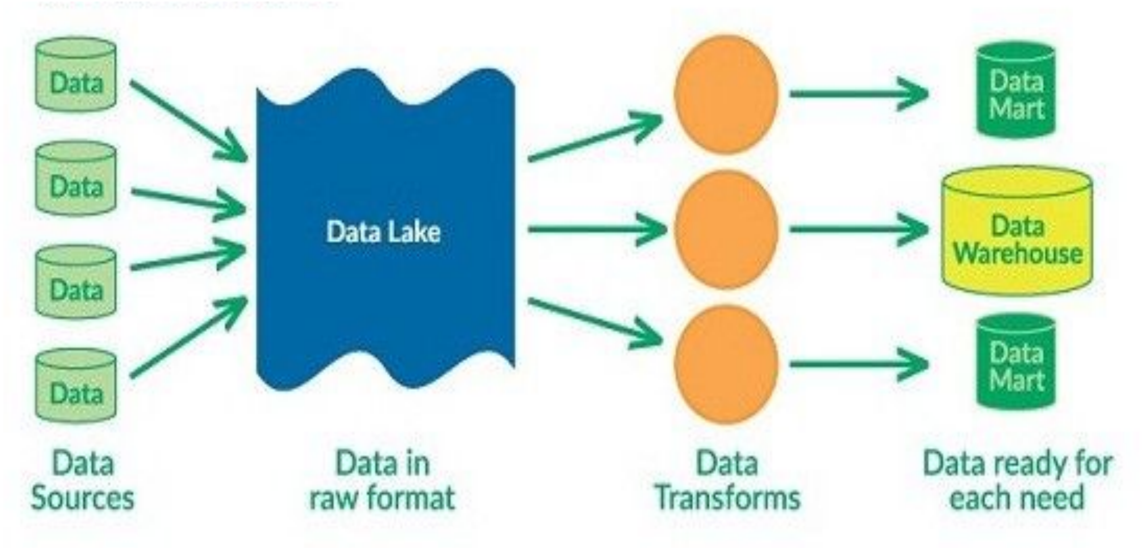
Agenda

- Towards Delta
  - Traditional Data Lakes
  - Lambda architecture
  - Current challenges

- Delta Lake
  - Why delta?
  - Architecture
  - Delta tables

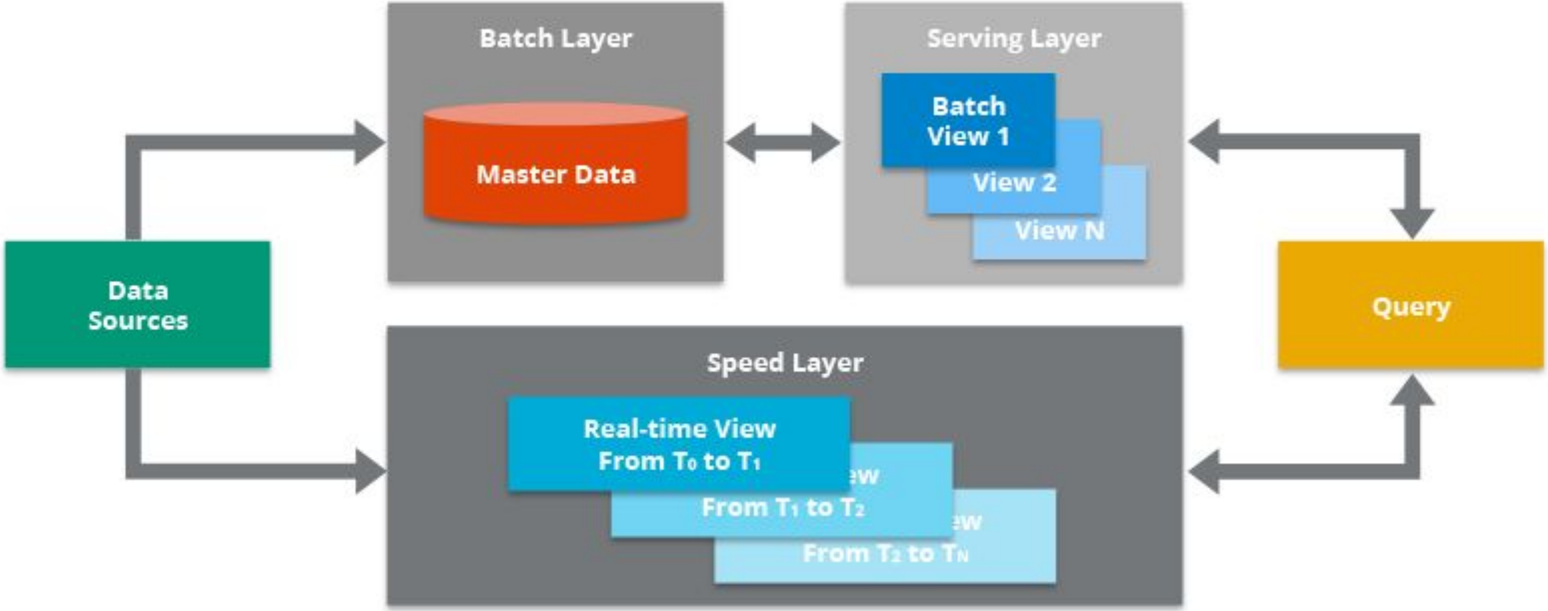
- Features of Delta Engines
  - ACID Transactions
  - Schema enforcement & evolution
  - Time Travel
  - Performance Optimization, Zorder etc

Traditional Data Lake (Handling Volume)



1. Input - Data from many sources.
2. Collected in a data lake.
3. Processed using some distributed processing engine.
4. Served for analysis.

Lambda Architecture (Handling Velocity)



https://hazelcast.com/wp-content/uploads/2020/04/19_Lambda.png

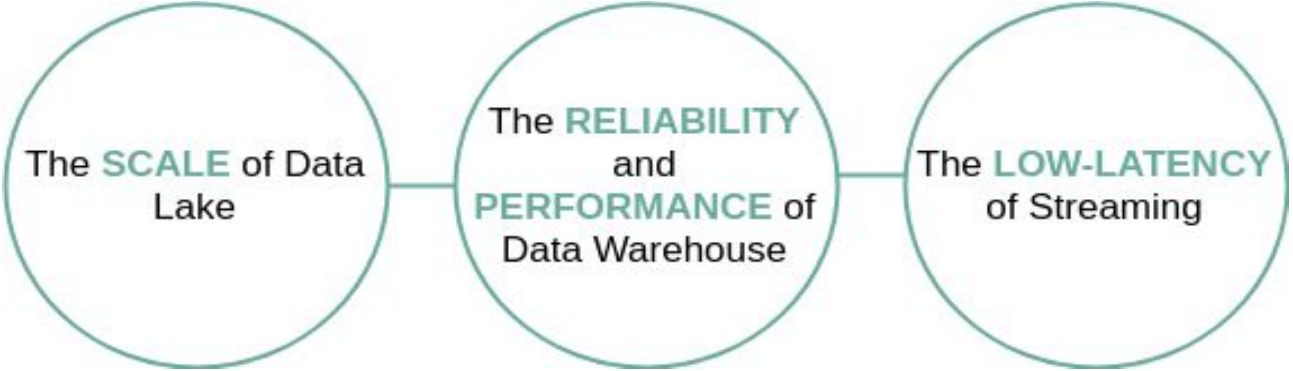
Challenges with current architecture

1. **Query Performance**
    - ETL process can add latency
    - With growing scale, query becomes slower.

2. **System Complexity**
    - Unifying stream and batch doubles infra cost.
    - Low level Code intervention needed to sync two pipeline data.
    - Need to reset pipelines for stream failures.

3. **Data Reliability**
    - Failed jobs can corrupt data
    - Schema changes can break joins, aggregates, transformations.
    - Hard to update/ delete data
    - Concurrent access suffer inconsistent query results.

Delta to the rescue!

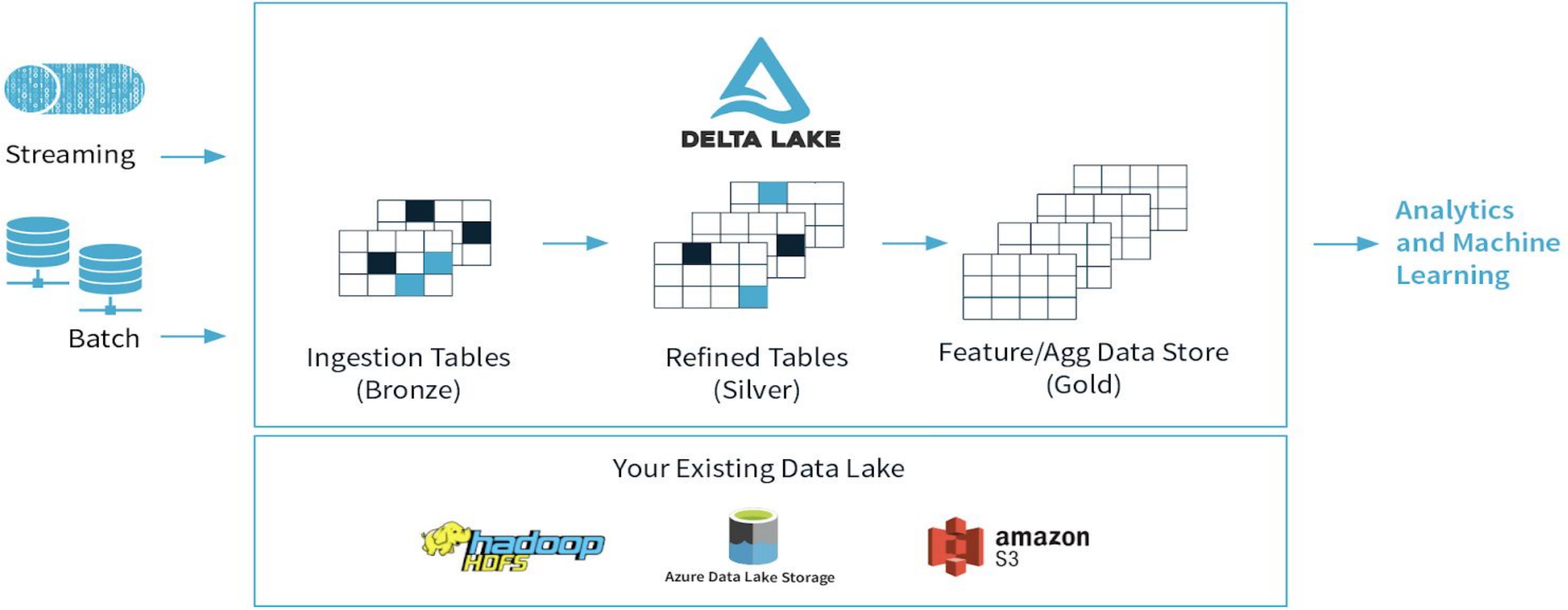

The SCALE of Data Lake
The RELIABILITY and PERFORMANCE of Data Warehouse
The LOW-LATENCY of Streaming

https://blog.knoldus.com/why-databricks-delta-called-as-unified-data-management/

- Delta stores data in versioned **Apache Parquet format.**
- Uses **Delta Transaction log protocol** to provide consistency.
- Provide Automatic **data indexing**.

*Incoming data is processed as “**delta**” records rather than the append-only new records.*

Delta Lake Architecture

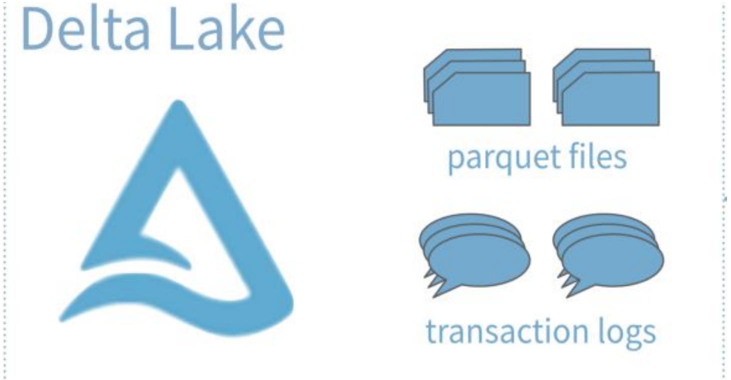


https://databricks.com/blog/2019/08/14/productionizing-machine-learning-with-delta-lake.html

Delta Table

To provide ACID guarantees we require the storage system to provide the following.

1. **Atomic visibility:** There must be a way for a file to be visible in its entirety or not visible at all.
2. **Mutual exclusion:** Only one writer must be able to create (or rename) a file at the final destination.
3. **Consistent listing:** Once a file has been written in a directory, all future listings for that directory must return that file.



https://github.com/delta-io/delta/blob/master/PROTOCOL.md

Storage systems do not necessarily provide all of these guarantees out-of-the-box.

Delta Lake transactional operations typically go through the **LogStore API** instead of accessing the storage system directly.

Delta Lake Features

Delta Lake is an open source storage layer that brings reliability to data lakes.

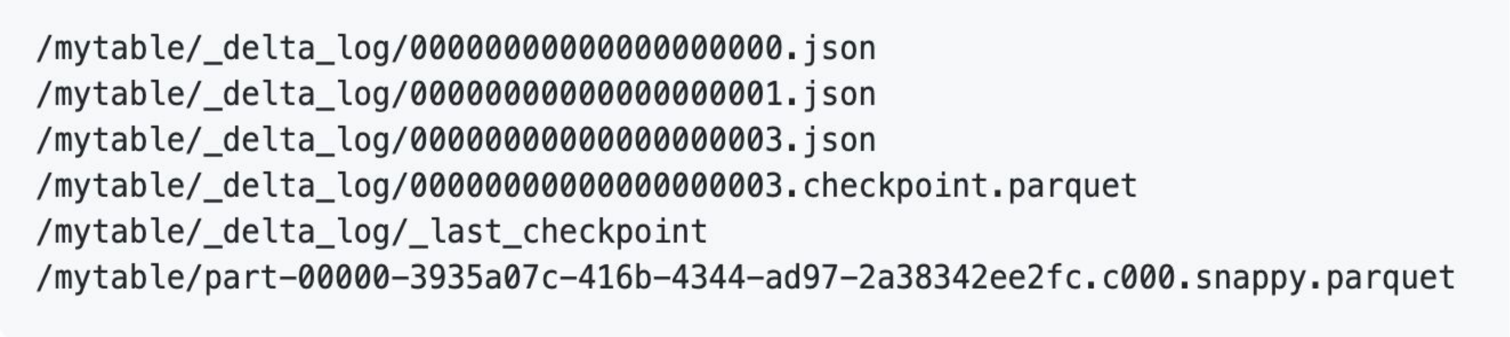
1. **ACID transactions**
2. **Scalable metadata handling**
3. **Time travel/ Data versioning**
4. **Streaming and batch unification**
5. **Schema enforcement**
6. **Schema evolution**
7. **Updates and deletes**
8. **100% compatible with Apache spark**

Dive into Delta Lake!

1. ACID Transactions Demo
2. Schema Enforcement and Evolution Demo

Demo Summary

1. Delta table is a **single serial history of atomic versions**.
2. The state of a table at a given version is called a **snapshot**.
3. Delta Lake uses ***optimistic concurrency control*** for concurrent read-write access.



```
/mytable/_delta_log/00000000000000000000.json
/mytable/_delta_log/00000000000000000001.json
/mytable/_delta_log/00000000000000000003.json
/mytable/_delta_log/00000000000000000003.checkpoint.parquet
/mytable/_delta_log/_last_checkpoint
/mytable/part-00000-3935a07c-416b-4344-ad97-2a38342ee2fc.c000.snappy.parquet
```

Data reliability by Delta

- **ACID Transactions** : “all or nothing” transactions
- **Snapshot Isolation** : multiple readers & writers support
- **Schema Enforcement**: Delta provides schema and prevent writes that do not align
- **Schema Evolution**: via merge schema or overwrite schema as needed.
- **Exactly Once**: Delta employs checkpointing to ensure data is not missed or repeated.
- **Upserts and Deletes support**: Spark tables are write once, modifications need to be done explicitly, while delta has implicit support for them.

Data Versioning

We have seen Delta Transactions & its Logs. Now, lets see how to use them for data versioning?

- Query older snapshot of delta table
- Revert to earlier versions of data for audits
- Trace transactions on the delta lake
- Reproducible machine learning experiments

https://www.thesun.co.uk/tech/8624077/time-machine-experiment-quantum-physics-electrons/

Performance Optimization

So far, delta lake supported us with ACID, data versioning, schema enforcement, etc.

But,

Is there any downside of this architecture?

1. **Query times?**
    a. Delta scans so many files
2. **Storage?**
    a. Delta maintain all versions of data

Now, lets see how delta helps improve query times.

Optimize via File Management

Data Compaction or Bin-Packing

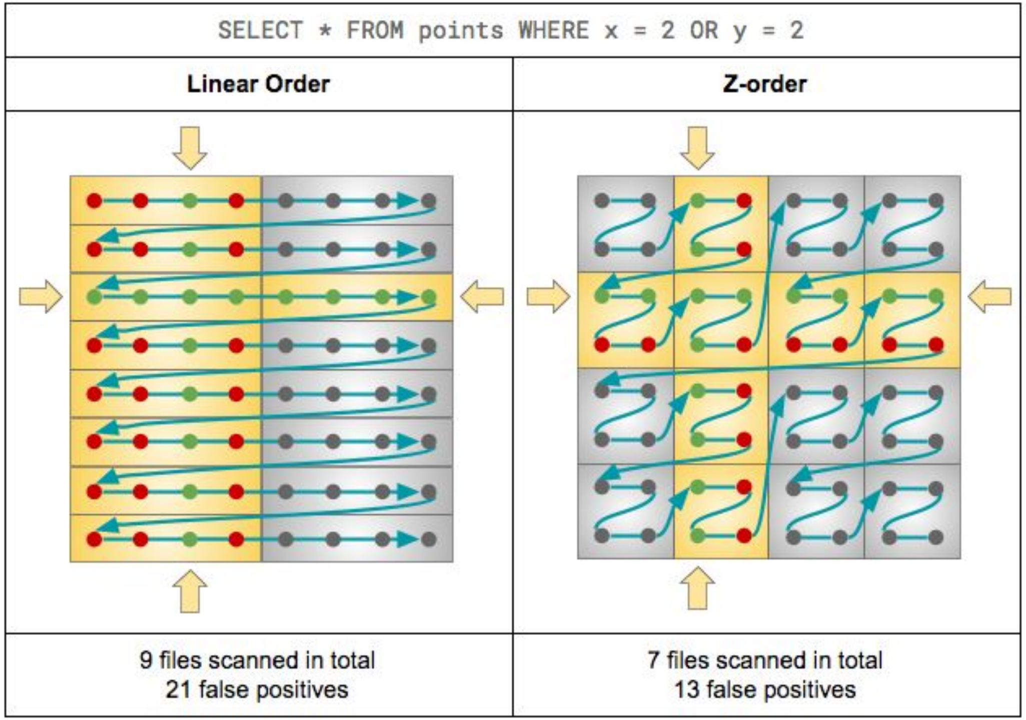
- Coalesce small files into larger ones.
- Produce evenly-balanced data files w.r.t their size on disk.
- No data related changes to the table.
- Idempotent.

Data Skipping

- Automatically collect statistics when you write data into a Delta table.
  - Parquet metadata supports this.
- Improve query execution time by skipping unrelated files.

For best results, apply **Z-Ordering**. ()

Z-ordering (multi-dimensional clustering)



```
SELECT * FROM points WHERE x = 2 OR y = 2
```

Linear Order	Z-order
9 files scanned in total 21 false positives	7 files scanned in total 13 false positives

- **Dot** = data points
- **Box** = data files; in this example, we aim for files of 4 points each
- **Yellow boxes** = data file that's read for the given query
- **Green dot** = data point that passes the query's filter and answers the query
- **Red dot** = data point that's read, but doesn't satisfy the filter; "false positive"

Image Reference: https://akashrehan.wordpress.com/2019/07/11/anatomy-of-databricks-delta-lake/

vacuum

- Remove files older than the retention threshold. Default is 7 days.
- Deletes only ***data files, not log files.***
- Not triggered automatically.
- Ability to time travel back to a version older than retention period is lost after running vacuum.

Delta Cache

- Accelerates data reads by creating copies of remote files in nodes’ local storage.
- Supports only Parquet files, not other storage formats.
- Stores data entirely on the local disk.
- Any stale entries are automatically invalidated and evicted from the cache.
- uses efficient ***decompression algorithms*** and outputs data in the optimal format

References

- https://docs.delta.io/latest/delta-batch.html#language-scala - Demo
- https://docs.microsoft.com/en-us/azure/databricks/delta/
- https://akashrehan.wordpress.com/2019/07/11/anatomy-of-databricks-delta-lake/
- https://blog.knoldus.com/databricks-delta-architecture/
- https://github.com/delta-io/delta

Thank You

Divya Dua
Data Engineer
divya.dua@thoughtworks.com

Nisha Kumari
Data Engineer
nishak@thoughtworks.com

ThoughtWorks®