Starburst

Not Your Father's Data Lakehouse

Building with Trino and Apache Iceberg

March 26, 2024 @ Data Council

Jack Klamer - Senior Software Engineer @ Starburst
Monica Miller - Devrel @ Starburst

Scan for a Trino and Iceberg cheat sheet

Workshop Agenda

Quick intro then hands-on lab

- Evolution of a data lakehouse (*the 3 min version*)
- Picking your components
- Building a data lakehouse
- Hands-on workshop!

Scan for a Trino and Iceberg cheat sheet

Starburst

Evolution of the data lakehouse

How did we get here?

Starburst

Data Architecture Evolution

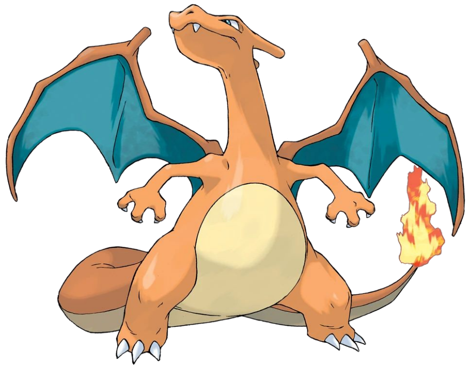
Data Warehouse

Charmander

Data Lake

Charmeleon

Data Lakehouse



Charizard

Starburst

The Data Warehouse

Popularized in the 90's to provide a 360 degree view

The Good

- Integrates siloed RDBMS's into one "centralized" location
- Simple & reliable analytical querying
- Data audit, governance and lineage
- Great for small amounts of data

The Bad

- Inability to store unstructured data
- Lack scalability and flexibility
- Tightly coupled storage and compute
- Expensive, proprietary hardware and software (*creating vendor lock-in*)

The Data Lake

Born out of the internet age and big data boom

The Good

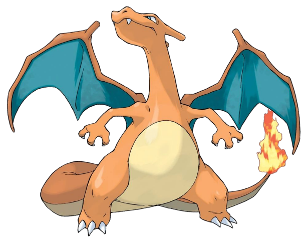
- In 2006, Apache Hadoop emerges so unstructured data can be processed at a scale previously imaginable
- Shift toward parallel processing
- Capitalize on low cost object storage
- Allows for greater flexibility (schema on read)

The Bad

- Inability to support transactions, updates, or modifications
- Difficult to get top tier performance
- Lack of data quality and inconsistent data formats
- Insufficient data lineage and limited data discoverability

Starburst

The Data Lakehouse



Applying data warehouse principles to the data lake

- Utilize the ***separation of storage and compute*** to apply the reliability, performance, data quality of the data warehouse to the openness and scalability of the data lake
- ***Increased performance and scalability*** through the use of indexing and caching via your query engine (**Trino**) and modern table formats
- Tackle ***unstructured, semi-structured, and structured*** analytical data all in a data lakehouse - creating a place for AI/ML & BI use cases alike

Starburst

Picking your components

Trino is the best query engine ever

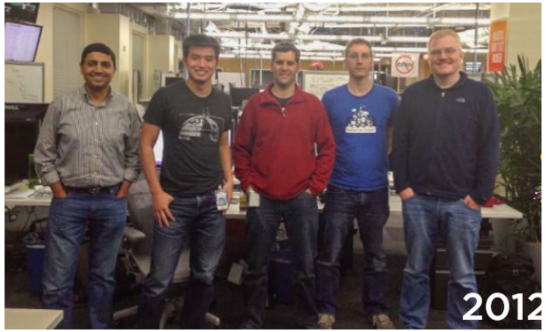
Starburst

The data accessibility problem

Data practitioners faced the same challenges at Facebook in 2010

- Facebook created Hive to query terabytes of data in Hadoop using SQL
- Data scientists attempted to query massive object stores, but performance was too slow
- Data consumers were limited by the number of queries they could run — often ***fewer than 10*** in one day

Enter Trino (Presto)



A new open source query engine designed for speed

Trino (*formerly known as Presto*) is a fast distributed SQL query engine designed to query large data sets distributed over one or more heterogeneous data sources.

- Harnesses the power of distributed computing
- Separates compute from storage
- ANSI SQL compliant

https://trino.io

Starburst

Most Common Uses *(other than powering data lakehouses)*

Interactive data analytics

Enter a SQL query for Trino to process and return results as quickly as possible.

- Query large amounts of data
- Test hypotheses
- Run A/B testing
- Build visualizations

High performance data lake analytics

Trino enables users to run SQL based analytics on HDFS/Hive and cloud object storage

- Run petabyte scale analytics
- Scale and performance benefits

Federated analytics

Create a single point of access by using Trino to query disparate data sources.

- Object storage
- Relational systems
- Streaming systems
- NoSQL systems

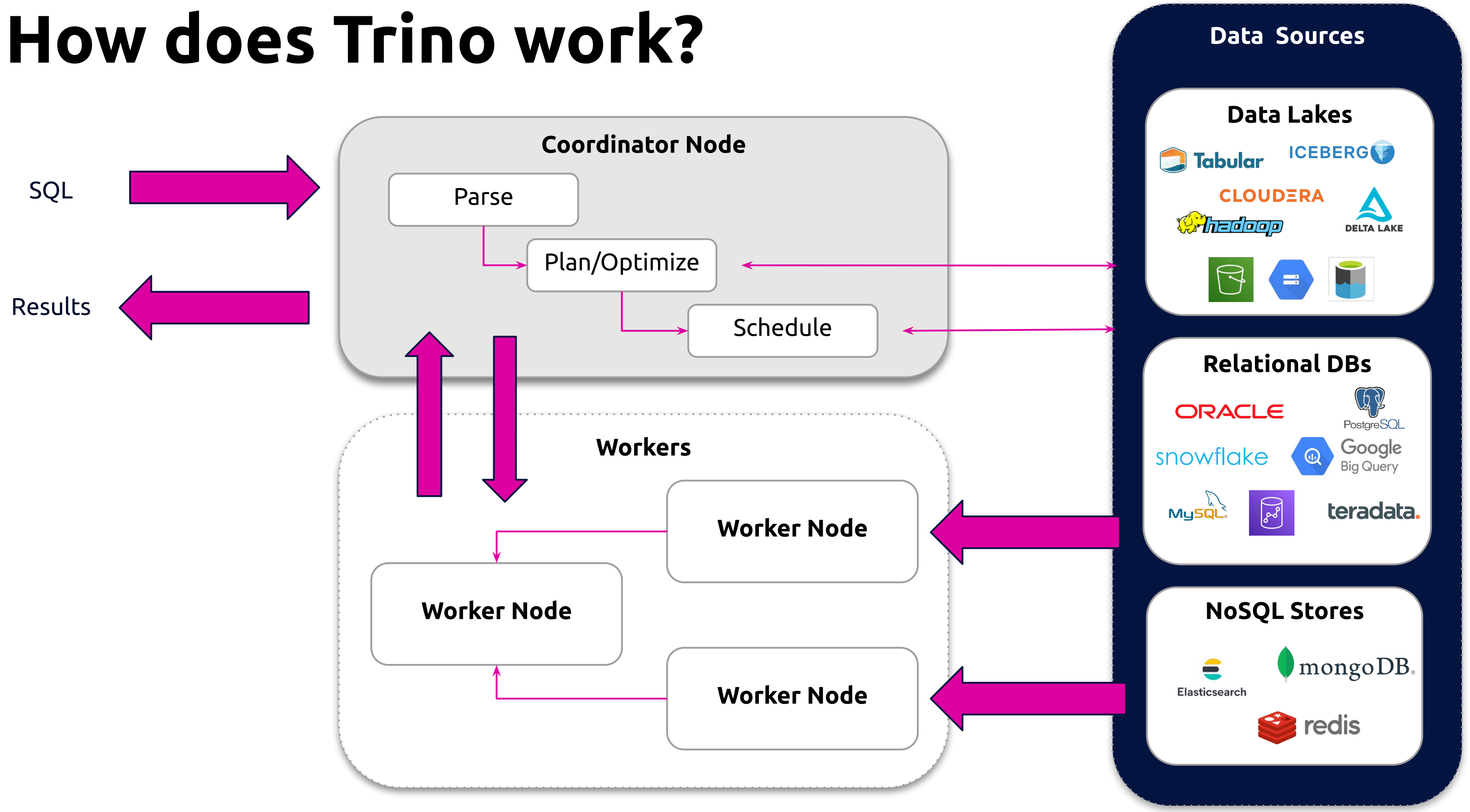
Batch ETL processing

Run resource intensive ETL processes in batches without fear of failure with Trino.

- Use SQL with every data source
- Work with numerous data sources and targets all in the same system
- Ensure speed and reliability

Starburst

How does Trino work?



https://trino.io

Starburst

Picking your components

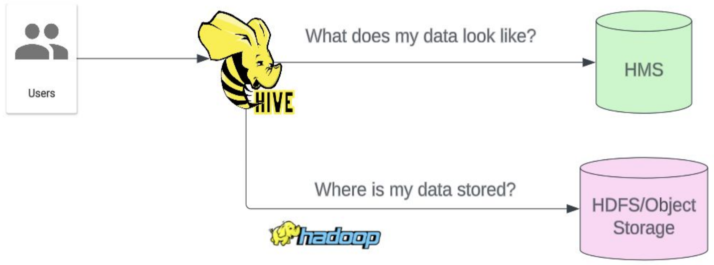
Iceberg is the industry standard table format

The Challenges of the invisible Hive "spec"

Hive has been critical for the evolution of SQL querying in distributed systems

Partitioning based on column names at the end of the table which match directory names on the file system (*users must know this*)

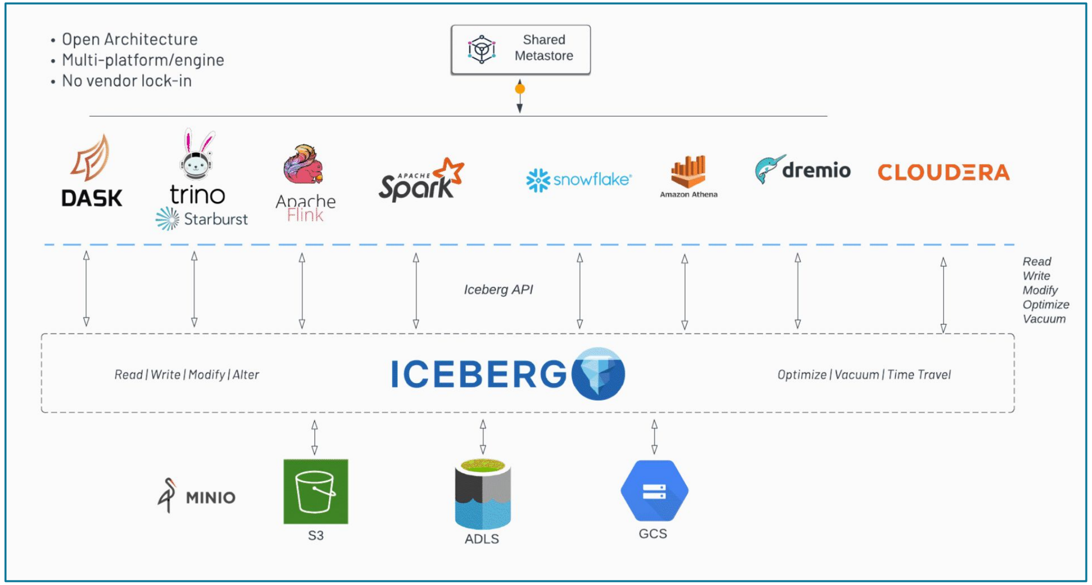
- Rigid partitions
- Partial schema evolution
- Not optimized for object storage
- Need list + scan all files in a folder
- Transactional/ACID has always been squirrelly (inconsistency, correctness issue)



Apache Iceberg

- Created by Ryan Blue & Daniel Weeks at Netflix in 2017
- Solve the challenges of performance, data modification and schema evolution in the lake
- Uses open data concepts (orc, parquet, avro) and architecture

Multi-Engine Platform



Iceberg: lake choice + warehouse behavior

SQL behavior

- Schema and layout evolution
- Hidden partitioning

Modern warehouse SQL

- MERGE
- UPDATE
- DELETE
- Time travel (VERSION AS OF)

Iceberg should be invisible

Avoid unpleasant surprises

- No zombie data
- Performance is not mysterious
- Reduced metastore reliance

Doesn’t steal attention

- Fast metadata operations
- Automate the boring stuff
- Fix problems without migration

Optimistic Concurrency

- Allows multiple writes simultaneously, checks for conflicts before final commit

Universal open standard

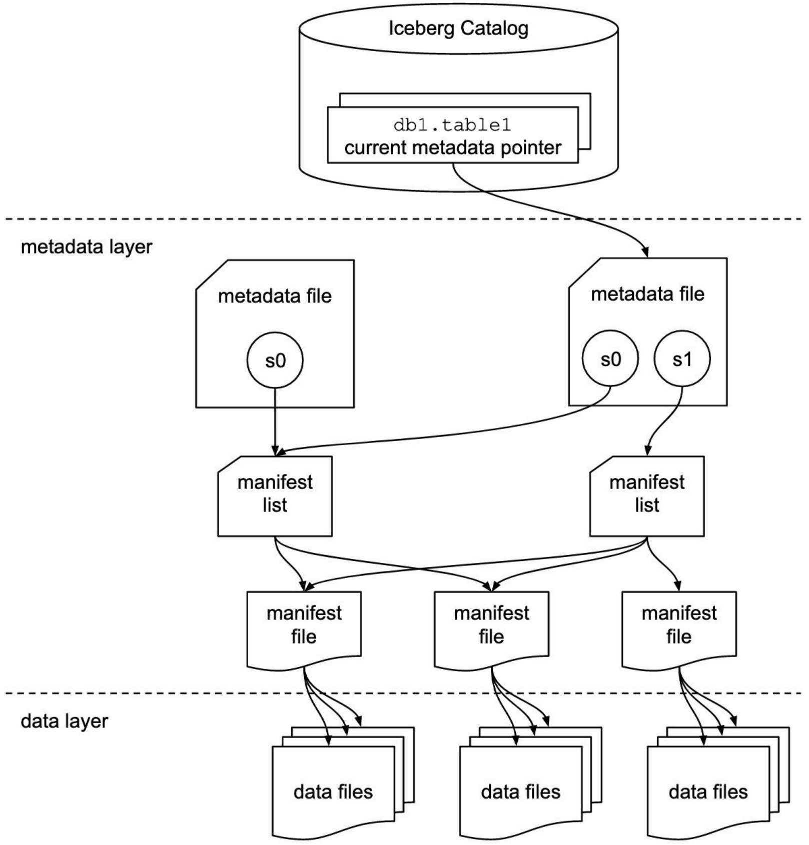
Architecture

Comprised of a **hierarchy of metadata files** to accommodate constant changes to a table (insert, delete, update, schema migration, partition changes).

Think of a **database transaction log but using an object store for the storage.**

Metadata:

- **Iceberg catalog** (HMS/Glue/JDBC) - Stores the file path for the "current" metadata file.
- **Metadata file** (json) - Stores information about table (schema/partition/etc) at a given point in time and details + pointers to snapshots (manifest list).
- **Manifest list** (avro) - Contains statistics for a collection of files that represent a single snapshot.
- **Manifest file** (avro) - List of data files (orc, parquet, avro), pruning by partition and column stats.



Building a data lakehouse

Starburst

Open Data Lakehouse Benefits

Data Warehouse Benefits

- ACID transactions
- Fined grained access control
- Data quality
- High performance and concurrency
- Highly curated data
- Typically proprietary systems
- Best for business intelligence use cases

Data Lake Benefits

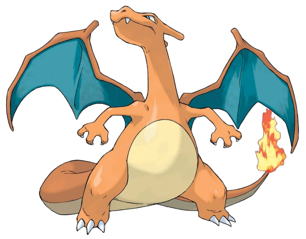
- Petabyte scale
- Cost efficient
- Open formats
- Separation of storage & compute
- Structured and unstructured data
- Best for data science and data engineering use cases

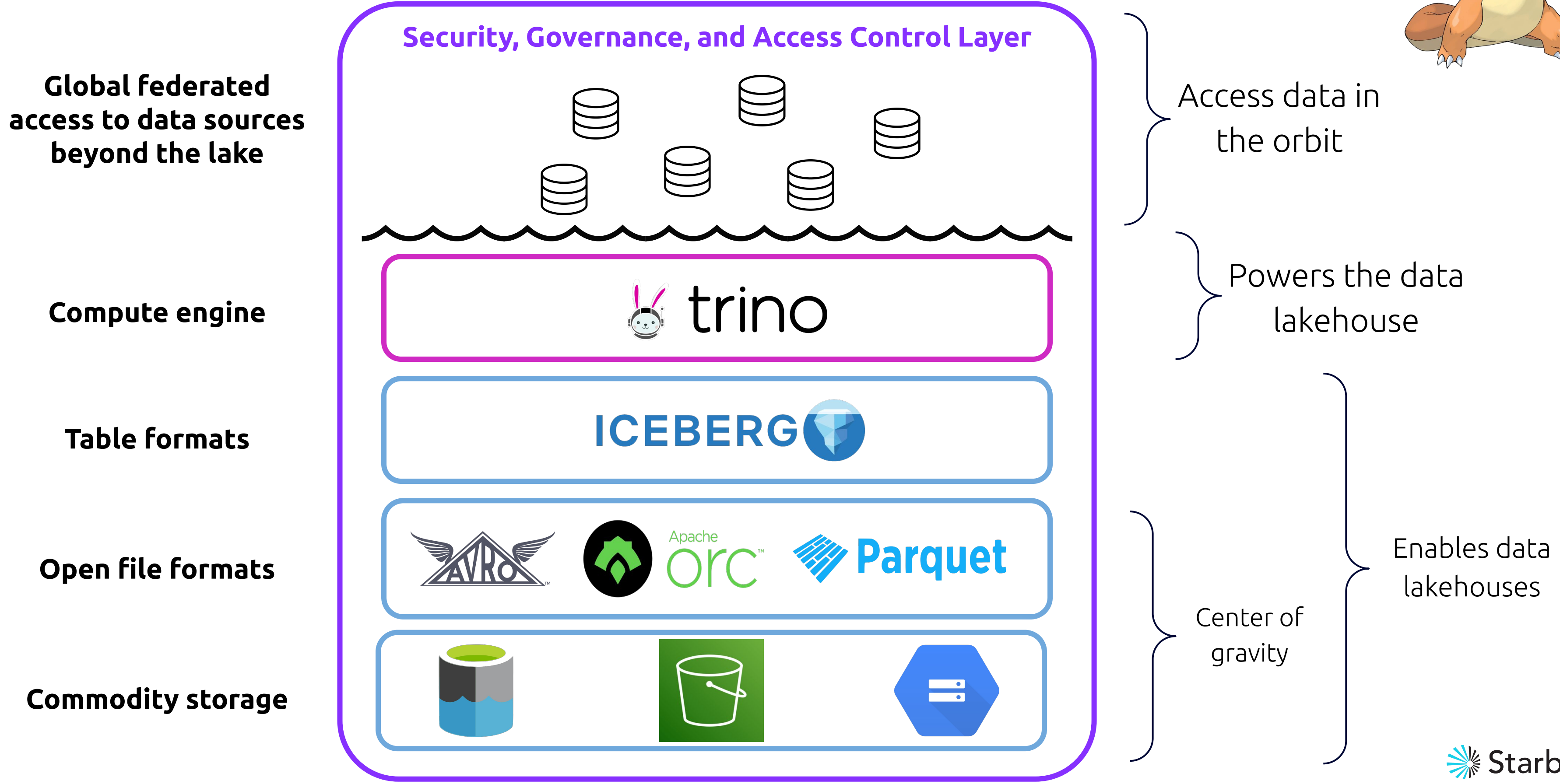
Lakehouse = the doodle of data architecture

Apply data warehouse principles to the data lake of your choice

Starburst

The Open Data Lakehouse





Starburst

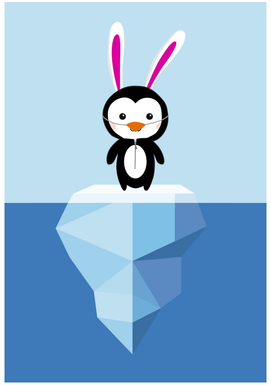
Hands-on Workshop

Starburst

References

Starburst

Iceberg + Trino = 💖



- Trino Community Broadcast - 40: Trino's cold as Iceberg! - Sep 8, 2022
- Introduction to Apache Iceberg in Trino
- Iceberg Partitioning and Performance Optimizations in Trino
- Apache Iceberg DML (update/delete/merge) & Maintenance in Trino
- Apache Iceberg Schema Evolution in Trino
- Apache Iceberg Time Travel & Rollbacks in Trino
- Building Reporting Structures on S3 using Starburst Galaxy and Apache Iceberg
- Near Real-Time Ingestion For Trino (with Flink and Iceberg)

Starburst

Use Cases

- Netflix Introduction and its origins at Netflix
- Apple Usage Iceberg, Trino and Spark - Iceberg contribution - Trino Summit 2022
- Airbnb Upgrading Data Warehouse Infrastructure at Airbnb (from Hive to Iceberg)
- Stripe Inspecting Trino on Ice
- Expedia A short introduction to Apache Iceberg
- SK Telecom Journey to Iceberg with SK Telecom - Trino Summit 2022

Starburst

Thank you!

Starburst