F1 Lightning
HTAP as a Service

Preetham Srinivasa Kikkeri

November 20th

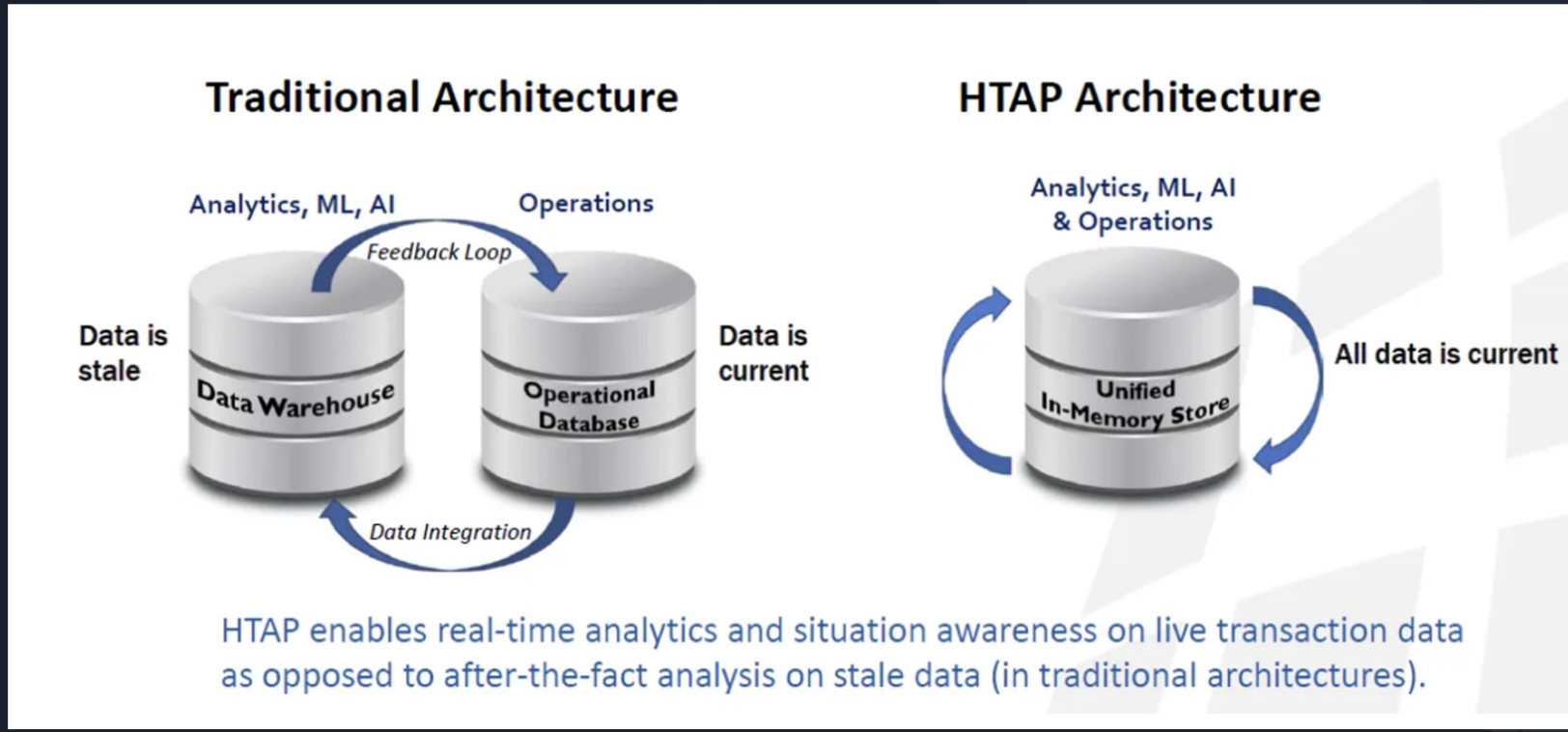
HTAP Intro

Hybrid transaction/analytical processing(HTAP), a term created by Gartner

As defined by Gartner:

Hybrid transaction/analytical processing(HTAP) is an emerging application architecture that "breaks the wall" between transaction processing (OLTP) and analytics (OLAP). It enables more informed and "in business real time" decision making.

Instantly run analytical queries on fresh transactional data



HTAP enables real-time analytics and situation awareness on live transaction data as opposed to after-the-fact analysis on stale data (in traditional architectures).

HTAP Solutions

1. **Single System for OLTP and OLAP**

   Systems that use hybrid row-wise and columnar data organizations tend to have better performance than those using a single data organization for both ingestion and analytics.

1. **Separate OLTP and OLAP Systems**

   2.1 Shared Storage

   Require modifications to the OLTP system - systems in this category are usually developed by the OLTP systems themselves to leverage an existing analytical query engine to accelerate analytic queries.

   2.2 Decoupled Storage

   No modifications can be made to the OLTP storage

   F1 Lightning is different from the single-system approach in that the analytic system is decoupled from the OLTP system because our users at Google cannot easily do a wholesale migration to an entirely new system.

HTAP Solutions

Many applications set up their HTAP architecture using loosely decoupled storage by maintaining a separate, offline ETL process.

Cons: Using offline ETL to ingest data into columnar file formats tends to suffer from high lag between the OLTP data and the OLAP copy.

F1 Lightning improves the data freshness via the integration with a Change Data Capture (CDC) mechanism and use of a **hybrid memory-resident and disk-resident storage hierarchy.**

Change Data Capture (CDC) incrementally extracts changes from source databases into a separate storage for analytics. These systems typically have improved change propagation delay and change replication efficiency over the traditional ETL approach that reimports the entire dataset

F1 Lightning - System Overview

1. **OLTP database**

   Source of truth - F1 DB, Spanner (row-oriented storage)

1. **F1 Query**

   Distributed SQL query engine

1. **Lightning**

   Maintains and serves read optimized replicas.

F1 Query

A federated engine that supports many different internal data sources, including F1 DB, Spanner, BigTable etc. In general, F1 DB and Spanner optimize for OLTP workloads by using efficient row-oriented storage to maximize write throughput.

Cons: analytical queries over these datasets by F1 Query are often suboptimal.

Alternative 1: F1 Query counteracts this by running distributed analysis queries with many workers

Cons: incurs substantial **computational resource costs** in order to provide reasonable latency.

Alternative 2: Copy F1 DB tables into ColumnIO files or other formats for further analysis.

Cons:

- Duplicate storage
- ColumnIO files do not support in-place updates - copies have to be periodically restated as a whole
- Staleness
- Different Queries

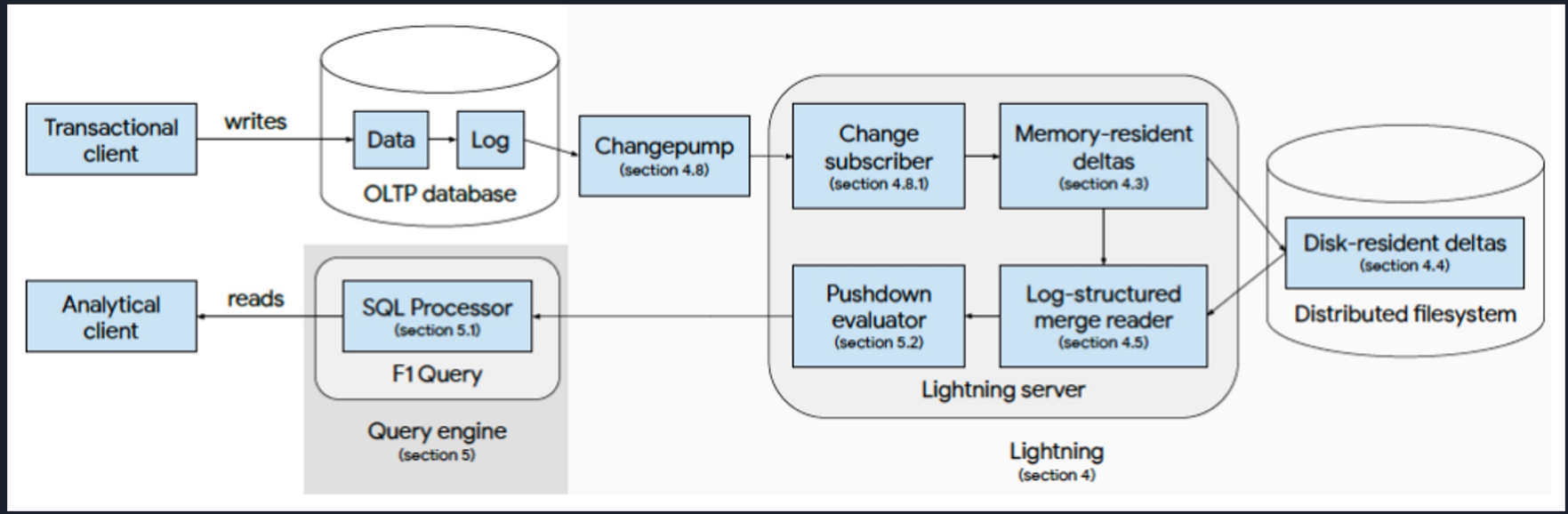
F1 Lightning

- Lightning - an HTAP system that replicates data from an OLTP database into a format optimized for analytical queries.
- Enable Lightning
  - table-by-table basis
  - entire database
- Lightning is a critical production service at Google, backing core Google products like AdWords and Payments

F1 Lightning - Benefits

- Improved resource efficiency and latency for analytic queries.
- Simple configuration and deduplication, avoid costly migrations.
- Separation of concerns - flexibility in the design of transactional storage systems
- Transparent user experience
- Data consistency and data freshness
- Data security

Architecture



- **Changepump** - uses the CDC mechanism to detect new changes and forwards those changes to partitions managed by individual Lightning servers, each of which maintain Log-Structured Merge (LSM) trees backed by a distributed file system.
- **Lightning servers** ingests these changes, transforms the change data from the row-oriented, write-optimized format used by the OLTP database into a column-oriented format optimized for analysis.

F1 Lightning

Lightning guarantees

Reads at a specific timestamp will produce results identical to reads against the OLTP database at the same timestamp.

F1 Query Accelerates Query performance

When a query requests to read F1 DB and Spanner at a particular timestamp, if that data is available in Lightning, the query will instead read from Lightning and benefit from improved performance without any explicit action on the part of the end user.

Read semantics

- **Supports multi-version concurrency control with snapshot isolation**

  All queries against Lightning-resident tables specify a read timestamp, and Lightning returns data consistent with the OLTP database as of that timestamp.

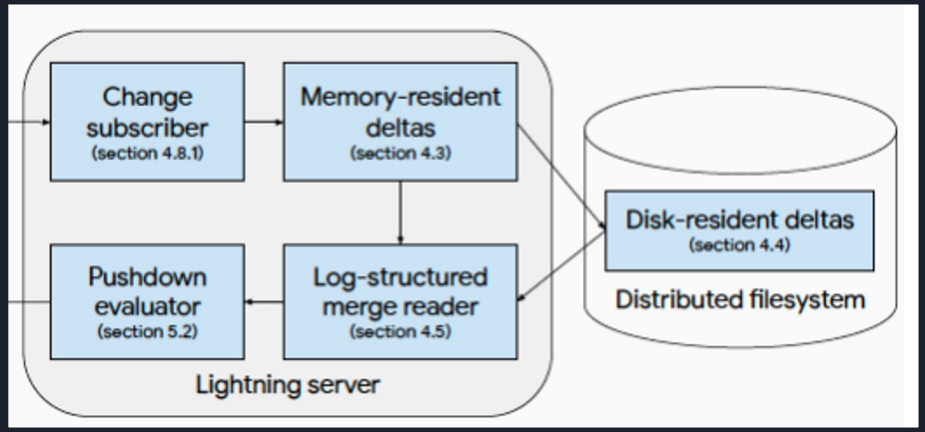
- **Maintains Queryable Window**

  Lightning has a single version snapshot of the database as of the minimum safe timestamp and a multi-version record from that point up to the maximum safe timestamp.

Tables and deltas

- Lightning stores data organized into Lightning tables.
  - Each Lightning table is divided into a set of partitions using range partitioning.
- Deltas contain partial row versions for their corresponding Lightning table.
- Within a delta, partial row versions are uniquely identified by <key, timestamp> pairs.
  - In order to support fast seeks to a specific timestamp, deltas are ordered by key ASC, timestamp DESC.

Memory resident Deltas



- When changes are ingested by Lightning, the resulting partial row versions are first written to a memory-resident, row-wise B-tree.
- Once data is written to a memory-resident delta, it is immediately available for querying.
- Periodically checkpoints memory-resident deltas to disk.
- In case of system failure, changes stored in memory may be lost. Lightning recovers from this state by replaying from the log of the OLTP database.

Disk resident Deltas

Stored in read-optimized columnar files.

Goal: Hybrid workloads - balance between range scan and point-lookup performance.

Each delta file stores:

- Data part

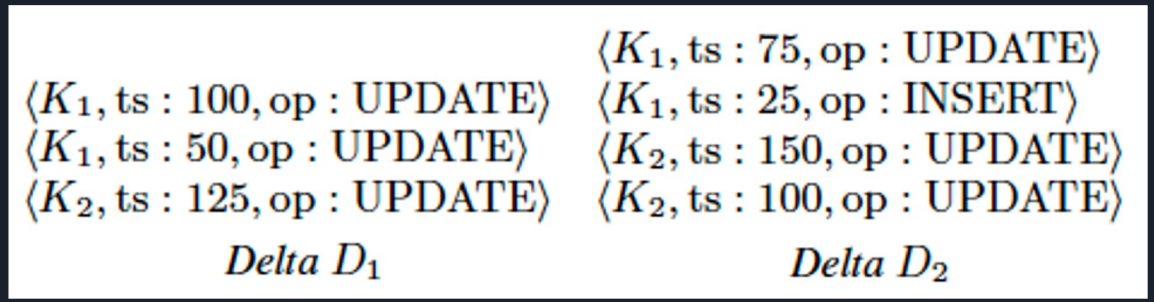
  Stores row versions in a PAX (Partition Attributes Across) layout where rows are first divided into row bundles then stored column-wise within a row bundle.

- Index part

  Contains a sparse B-tree index on the primary key.

Delta merging

- Read must merge deltas and combine row versions in order to form complete rows.
  - **Merging:** deduplicates changes in the source deltas and copies distinct versions to the new delta.
  - **Collapsing:** combines multiple versions of the same key into a single version. Lightning must take care that it collapses only complete histories with no holes.



Deltas are ordered by key ASC, timestamp DESC.

Can only collapse versions whose key and timestamp are less than than K2, ts : 125 in this round. This is because D1 may have additional versions for K2 whose timestamp is between 100 and 125, but this will not be determined until the next batch is read from D1.

Delta Compaction

Lightning runs periodic delta compaction operations to rewrite smaller deltas into a single larger delta.

1. **Active compaction**

   Performs delta compaction on **memory-resident** deltas

1. **On-Disk compaction**
   - Minor - compacts small and likely fresh deltas
   - Major - handles large and old deltas.

Change replication & Subscriptions

Changepump provides a unified interface across different transactional sources.

- Hides the details of individual OLTP databases from the rest of the system. For each supported transactional source, an adapter inside Changepump converts from that system's change data capture format into a unified format.
- Lightning maintains a subscription to Changepump for each partition. The subscription specifies the partition's table and key range, and Changepump is responsible for delivering those changes to the Lightning server. A single subscription may connect to multiple changeservers

Sharding

- Changepump servers are partitioned with the goal of roughly-equal partitions in terms of the **number of new writes** each partition sees
- Lightning servers are partitioned with the goal of roughly-equal partitions in terms of the total **number of rows** contained in each partition.

F1 Query Integration

Standardised on F1 Query as the interface for Lightning - All lightning reads are serviced by F1 Query

Transparent Query Rewrites

the entity issuing the query continues querying against the OLTP database without modifying their queries, and they may not even know that Lightning is in the picture.

Subplan Pushdown

This enables the F1 Query optimizer to consider a rich set of options for pushdown into Lightning servers

Thank You