WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CS 764: Topics in Database Management Systems

Lecture 12: HTAP

Xiangyao Yu
10/14/2025

Announcements

Submit project proposal by **Oct. 17**

- Create a new submission on Hotcrp (https://wisc-cs764-f25.hotcrp.com)

Cover the following aspects (in 1–4 pages)

- Project name
- Author list
- Background and motivation (why important? challenges?)
- Task plan (what will you do? key contributions?)
- Timeline

Recommend ACM format

- https://www.acm.org/publications/proceedings-template

Today's Papers — HTAP

How Good is My HTAP System?

Elena Milkai, Yannis Chronis, Kevin P. Gaffney, Zhihan Guo, Jignesh M. Patel, Xiangyao Yu
University of Wisconsin-Madison
{milkai,kpgaffney}@wisc.edu,{chronis,zhihan,jignesh,yxy}@cs.wisc.edu

ABSTRACT

Hybrid Transactional and Analytical Processing (HTAP) systems have recently gained popularity as they combine OLAP and OLTP processing to reduce administrative and synchronization costs between dedicated systems. However, there is no precise characterization of the features that distinguish a good HTAP system from a poor one. In this paper, we seek to solve this problem from the perspectives of both *performance* and *freshness*. To simultaneously capture the performance of both transactional and analytical processing, we introduce a new concept called *throughput frontier*, which visualizes both transactional and analytical throughput in a single 2D graph. The throughput frontier can capture information regarding the performance of each engine, the interference between the two engines, and various system design decisions. To capture how well an HTAP system supports real-time analytics, we define a *freshness* metric which quantifies how recent is the snapshot of the data seen by each analytical query. We also develop a practical way to measure freshness in a real system. We design a new hybrid benchmark called *HATtrick* which incorporates both *throughput frontier* and *freshness* as metrics. Using the benchmark, we evaluate three representative HTAP systems under various data size and system configurations and demonstrate how the metrics reveal important system characteristics and performance information.

CCS CONCEPTS

• **Information systems → Database performance evaluation**.

KEYWORDS

HTAP; throughput frontier; freshness score; benchmark

ACM Reference Format:
Elena Milkai, Yannis Chronis, Kevin P. Gaffney, Zhihan Guo, Jignesh M. Patel, Xiangyao Yu. 2022. How Good is My HTAP System?. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3514221.3526148

1 INTRODUCTION

Modern enterprises often have both *online transactional* (OLTP) and *online analytical processing* (OLAP) workloads, and they typically serve these workloads using separate databases. Maintaining multiple database instances, however, requires additional administrative effort and computational resources. In recent years, there is a growing demand for database systems that can serve both OLTP and OLAP workloads with performance comparable to more specialized systems. This approach, called *Hybrid Transactional and Analytical Processing* (HTAP), is now an active area of research.

Interestingly, even after a hugefl urry of papers in the HTAP field [4–6, 10, 11, 13, 15, 17, 18, 20–23, 32, 33, 37, 38, 43], there is no clear definition of what makes an HTAP system *good*, beyond its ability to support a wide range of mixed OLTP/OLAP workloads. As a result, it is difficult to characterize and compare the performance of HTAP systems. To address this limitation, we define a systematic approach to characterize the behavior of HTAP systems.

We propose two methods that define and measure HTAP performance. First, a method to characterize an HTAP system based on a performance-centric definition. Second, a method to measure the ability of an HTAP system to provide real-time analytics.

In practice, we rarely see pure OLTP or OLAP workloads; for example, TPC-E [1] has a mix of both OLTP and OLAP queries. Nevertheless, we build on these two extreme cases (pure OLTP and OLAP) and call the space in between the HTAP spectrum. Conceptually, for any point on the HTAP spectrum, a good HTAP system does not favor the OLTP or OLAP component at the expense of the other component. We capture the overall performance over the HTAP spectrum with a new concept called *throughput frontier*. This concept combines the performance of the OLTP and OLAP workload components. By visualizing the throughput frontier in a 2D chart, we can understand the global performance behavior of an HTAP system, as well as identify problematic areas.

A key concept in the HTAP domain is *freshness*. Loosely defined, the freshness of an HTAP system is a measure of delay with which updates to the database (from the OLTP component of the workload) are made visible to the OLAP queries. A more precise definition of freshness will be provided in Section 4.1. We also propose a method to measure *freshness* in practice for any HTAP system. The empirical value of *freshness* allows us to understand whether the HTAP system is able to provide real or near-real time analytics.

We present a new benchmark called *HATtrick*, which we use to validate the concept of throughput frontier and our proposed method to measure freshness. The HATtrick benchmark employs a family of systematically generated workloads. At one extreme is a purely transactional workload, and at the other extreme is a purely analytical workload. In between, the workload has a mix of transactional and analytical operations. For each workload, HATtrick measures and extracts the performance and freshness of the HTAP system at that "operating point". To run HATtrick, one executes a range of workloads as specified by the benchmark parameters.

Finally, we run the HATtrick benchmark on a number of database systems to understand their characteristics. Our results show that the shape of the throughput frontier reveals essential information about the ability of a database system to concurrently serve

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3526148

Hermes: Off-the-Shelf Real-Time Transactional Analytics

Elena Milkai
University of Wisconsin-Madison
milkai@wisc.edu

Xiangyao Yu
University of Wisconsin-Madison
yxy@cs.wisc.edu

Jignesh M. Patel
Carnegie Mellon University
jignesh@cmu.edu

ABSTRACT

Many modern applications require real-time analytics, where analytical processing (AP) workloads needs access to the latest data updates from a transactional processing (TP) engine. However, managing separate TP and AP engines across teams complicates achieving real-time analytics without switching to specialized HTAP systems. To address this challenge, we introduce *off-the-shelf real-time analytics*, a system design that leverages the existing TP and AP engines to provide (1) the latest transactional updates for analytical queries and (2) support for efficient *transactional analytics*–transactions that combine transactional logic and analytical queries within a single ACID transaction–at various isolation levels. We demonstrate this concept with a new service called *Hermes*, which acts as a middleware that merges log records with analytical reads without altering existing engines. Our evaluation utilizes two AP engines, *FlexPushdownDB* and *DuckDB*, with *MySQL* as the TP engine. Using the *HATtrick* benchmark and a new workload called *Transactional Analytics Workload* (TAW), we compare Hermes with the leading HTAP solution, *TiDB*. Our results indicate that Hermes performs comparably to current HTAP solutions for real-time analytics and surpasses them by 3× in transactional analytics performance.

PVLDB Reference Format:
Elena Milkai, Xiangyao Yu, and Jignesh M. Patel. Hermes: Off-the-Shelf Real-Time Transactional Analytics. PVLDB, 18(8): 2334-2347, 2025.
doi:10.14778/3742728.3742731

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/elenamilkai/Hermes_layer.git.

1 INTRODUCTION

Real-time data analytics is increasingly critical in modern applications [36, 39, 64] such as fraud detection, dashboarding, healthcare, and cluster monitoring. These applications require analytical processing (AP) to read fresh updates from the transactional processing (TP) engines in real time. It is commonplace for organizations to maintain multiple independent and heterogeneous TP and AP engines [57, 66]. These engines, potentially managed by distinct teams, cater to specific use cases. Providing real-time analytics in such an architecture is challenging due to the need for synchronization.

The state-of-the-art solution for achieving real-time analytics over fresh data is through hybrid transactional/analytical processing (HTAP) systems [25, 29, 31, 33, 37, 45, 50, 51, 54, 55, 61, 63, 65]. However, existing HTAP systems face a fundamental limitation: **Compulsory migration.** HTAP solutions tightly couple AP and TP for high performance and fresh queries. Consequently, organizations must transition from their current TP and AP engines and migrate all data functions into a new HTAP database. This incurs significant cost and management overhead [9, 71]. Moreover, this migration may not be feasible if the HTAP engine fails to support essential functions provided by the current TP or AP engines. Supporting all functions from diverse engines into a single HTAP system is a daunting challenge.

To address this limitation and build a system that achieves real-time analytics using the existing TP and AP engines already deployed in an organization, we must tackle two key challenges:

- **Challenge 1: Support for pluggable engines.** We need a solution that simplifies the integration of *most* TP and AP engines that meet certain specific requirements, acknowledging that it is challenging to generalize for all engines. Our goal is to minimize or, ideally, eliminate the need for modifications within these engines by creating a solution compatible with foundational principles across TP and AP engines, which we discuss in Section 3.2. The optimal approach would involve implementing real-time analytics functionality externally, reducing the need for extensive alterations to individual engines. This approach enhances adaptability and seeks a solution that can be generalized across a wide range of engines, while recognizing the limitations in achieving complete universality.
- **Challenge 2: Efficient True HTAP Transactions.** *True HTAP transactions* [57] are transactions in which transactional logic and analytical queries are processed within a single ACID transaction. This capability simplifies application development [30] and enables new application scenarios [41, 75], which we discuss in detail in Section 4.3. In this paper, we refer to this capability as *Transactional Analytics* and label such a transaction an *Analytical Transaction.* Achieving efficient execution of real-time transactional analytics within a decoupled architecture imposes significant challenges. While distinct specialized engines enable effective execution of each component of these transactions, they complicate the maintenance of data consistency and correctness according to the specified isolation level.

In this paper, we introduce *off-the-shelf real-time analytics*, a novel architecture designed to address these challenges as follows: **Solution to Challenge 1:** An off-the-shelf real-time analytics system is constructed using *existing TP and AP engines* with no or minimal modifications to them. The key insight is to introduce a new system layer between the database engines and the storage, which merges the transactional logs with the analytical reads for

Proceedings of the VLDB Endowment, Vol. 18, No. 8 ISSN 2150-8097.
doi:10.14778/3742728.3742731

Outline

Demand of real-time analytics

HATtrick benchmark

- Throughput frontier
- Freshness

Hermes

- Off-the-shelf real-time analytics
- Transactional analytics
- Evaluation

A Brief History of DB

1980s: Relational DBMS + transactions

1990s: Separation of TP and AP; nightly ETL

2000s: Optimized AP: column store, vectorized

2010s: Real-time analytics.

- Streaming systems
- Hybrid Transactional/Analytical Processing (HTAP)

The Need for Real-Time Analytics

Analytical queries depend on fresh transactional data available in real-time

Critical applications that demand timely insights

- Fraud detection, IoT analytics, interactive dashboards, healthcare, and cluster monitoring

Stale data risks poor decisions, delays, and lost opportunities

Outline

Demand of real-time analytics

HATtrick benchmark

- Throughput frontier
- Freshness

Hermes

- Off-the-shelf real-time analytics
- Transactional analytics
- Evaluation

Motivation

Hybrid Transactional Analytical Processing (HTAP) is active research area with a flurry of papers.

- Single system for TP and AP
- Separate TP and AP systems

No definition of a good HTAP system

Difficult to characterize and compare

Outline

Demand of real-time analytics

HATtrick benchmark

- **Throughput frontier**
- Freshness

Hermes

- Off-the-shelf real-time analytics
- Transactional analytics
- Evaluation

Throughput Frontier Metric

Input: (τ, α) pairs

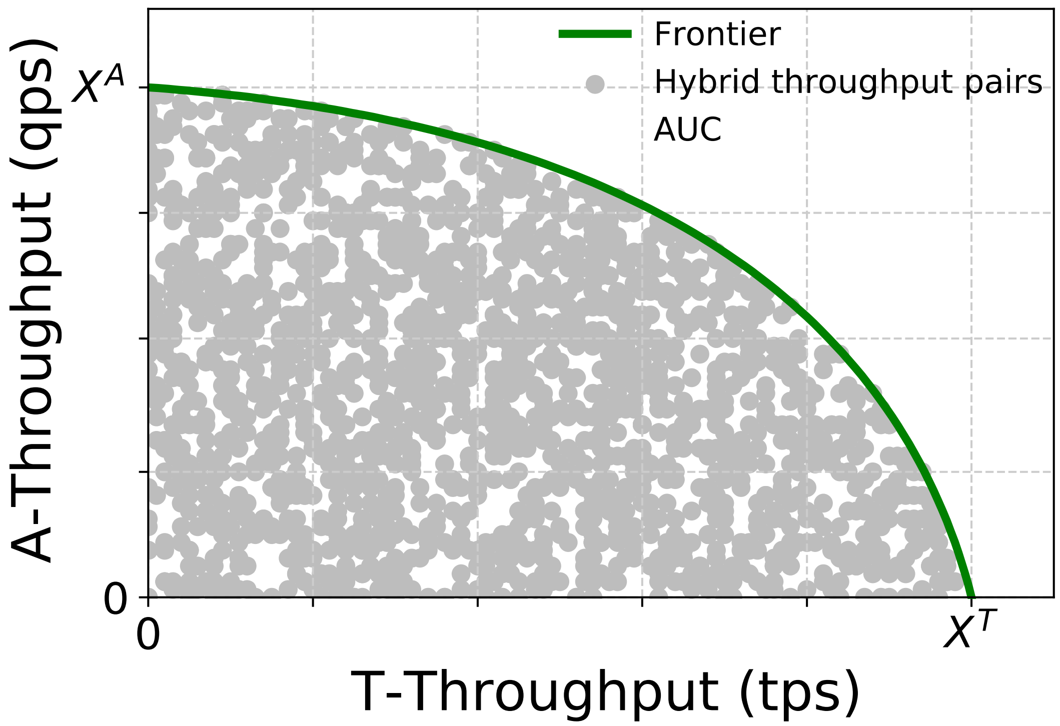
- τ transactional-clients
- α analytical-clients

Output: hybrid throughput $(x_τ, x_α)$

Sample all possible (τ, α) mixes

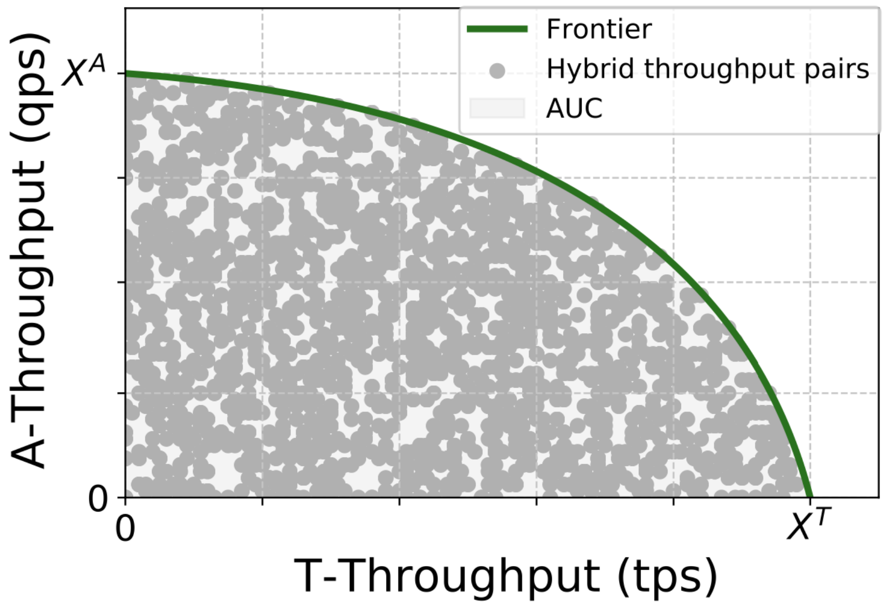
Map hybrid throughputs to 2D space

Max hybrid throughput values → throughput frontier

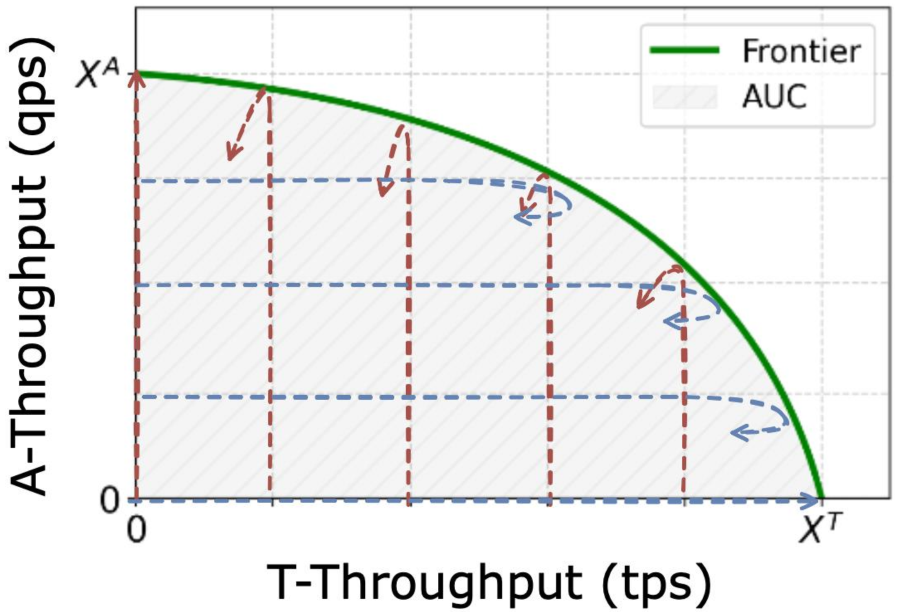


Throughput Frontier—Measure in Practice

Sampling Method



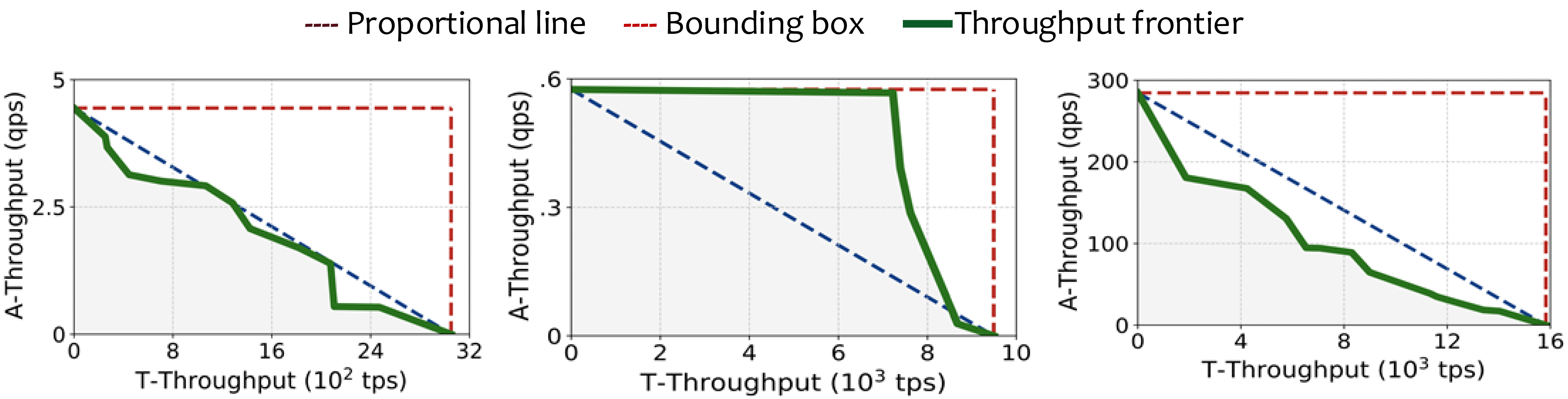
Saturation Method



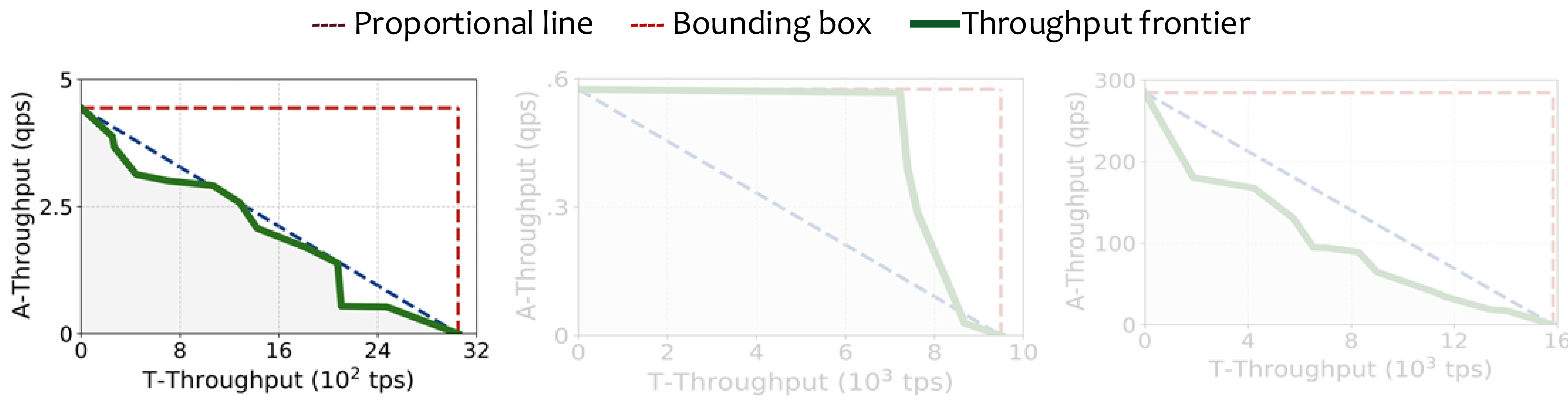
Throughput frontier: Maximum hybrid throughput across TP and AP engines

- Fixed-T lines: Fix the # of *T* clients and increase the # of *A* clients.
- Fixed-A lines: Fix the # of *A* clients and increase the # of *T* clients.

Patterns of Throughput Frontier



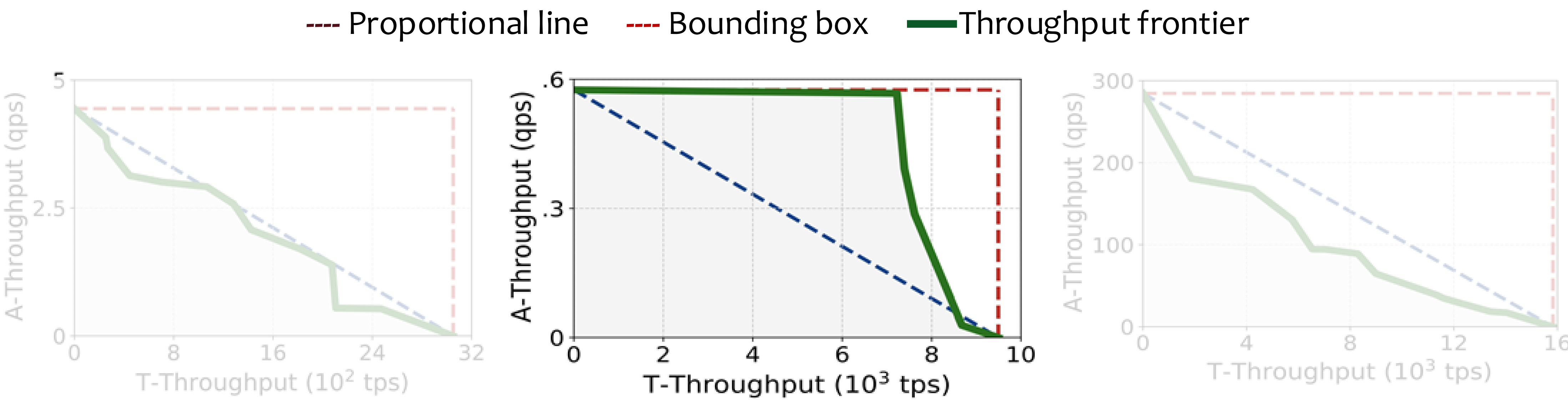
Patterns of Throughput Frontier



Pattern #1: **Close to the proportional line**

- Linear dependence between transactions and analytics
- Transactions and analytics share resources

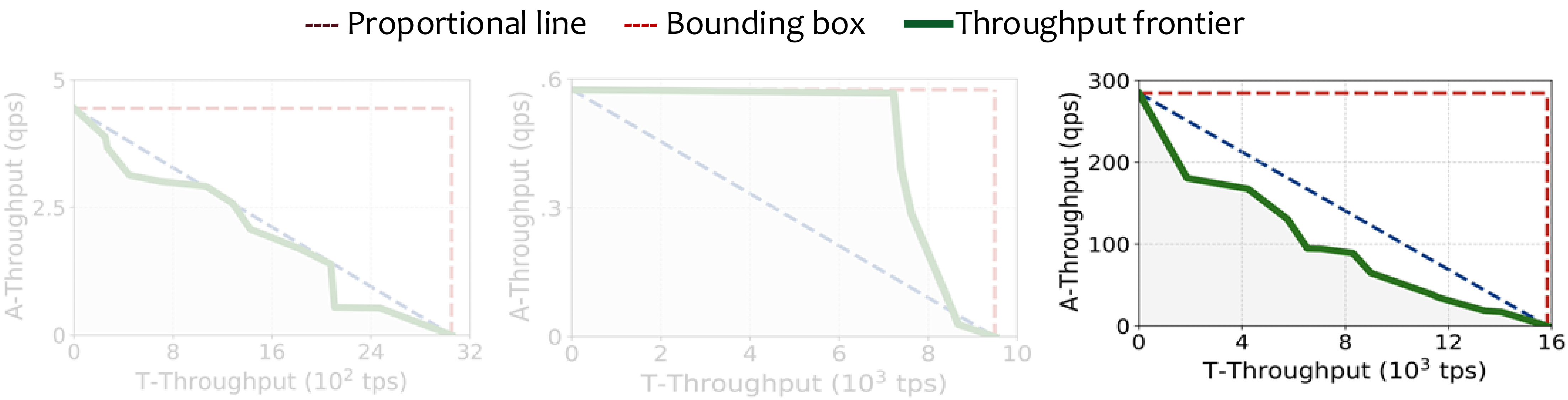
Patterns of Throughput Frontier



Pattern #2: **Above the proportional line, close to the bounding box**

- Independence between transactions and analytics
- Good performance isolation

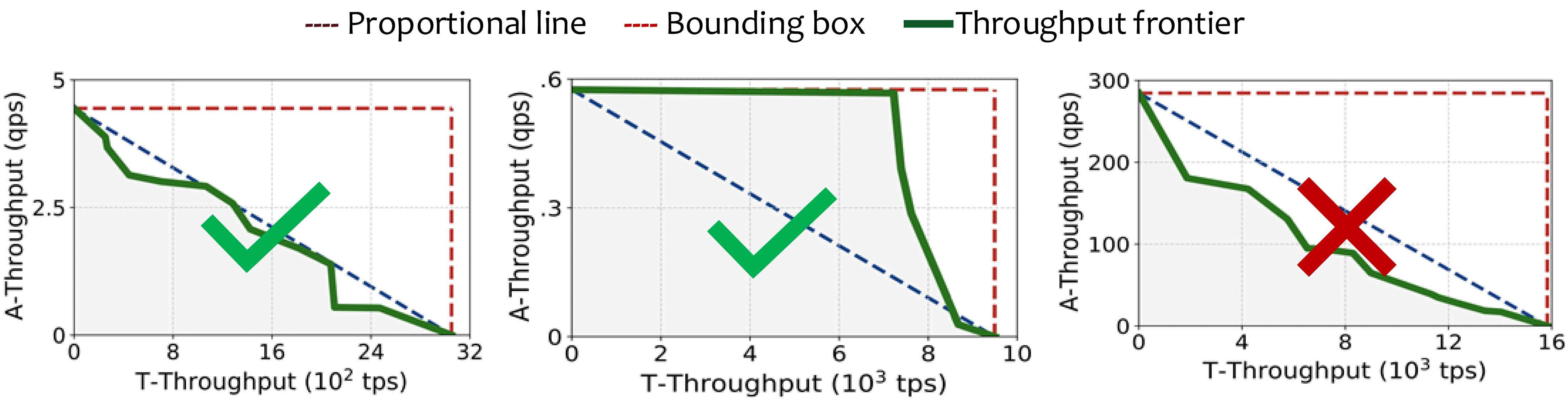
Patterns of Throughput Frontier



Pattern #3: **Below the proportional line, close to the axes**

- Interference between transactions and analytics
- Contention for resources

Patterns of Throughput Frontier



Outline

Demand of real-time analytics

HATtrick benchmark

- Throughput frontier
- **Freshness**

Hermes

- Off-the-shelf real-time analytics
- Transactional analytics
- Evaluation

Freshness Metric

Is the analytical query reading from a fresh snapshot?

Need a metric to capture recency of the query snapshot

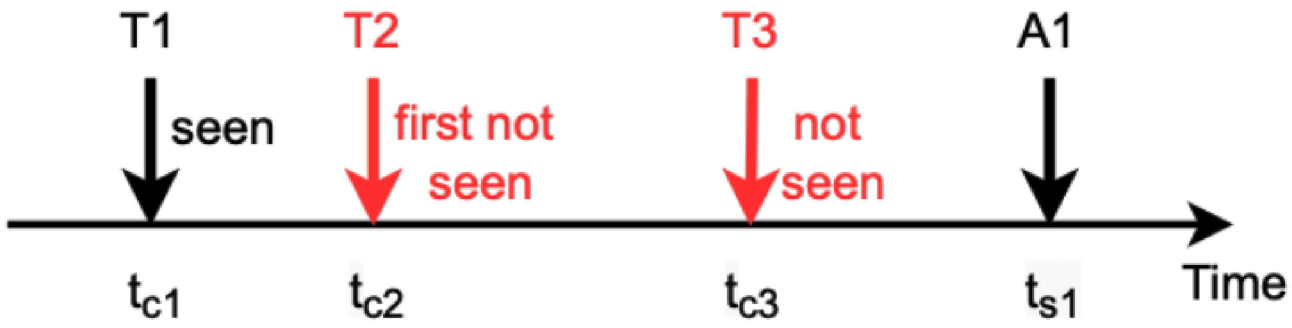
→ *freshness Metric*

Freshness Metric

$$f_{A_q} = max(0, t^{s}_{A_q} - t^{fns}_{A_q})$$

$t^{s}_{A_q}$: start time of analytical query A_q

$t^{fns}_{A_q}$: commit time of the **f**irst transaction **n**ot **s**een by A_q



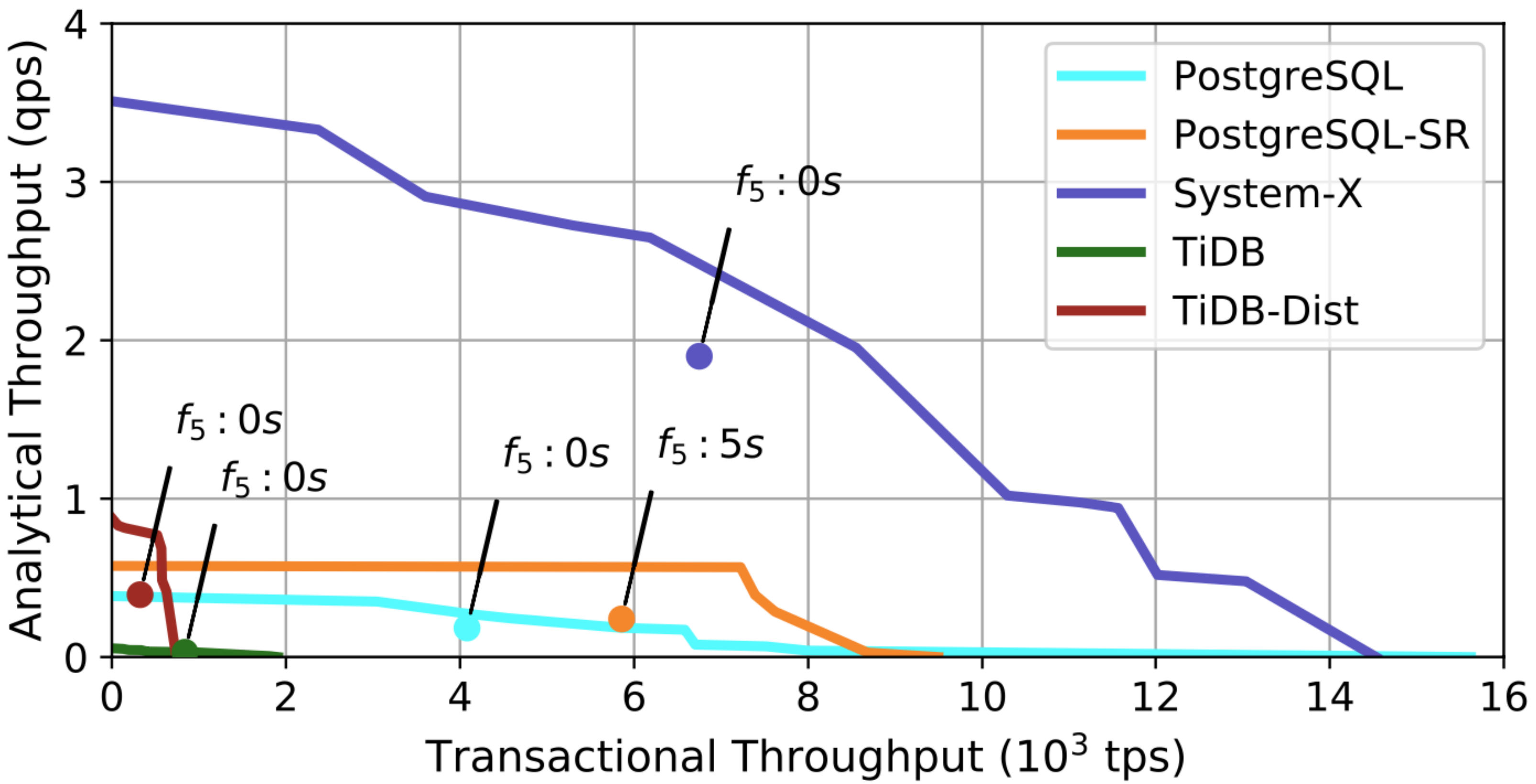
HATtrick Benchmark

Hybrid benchmark
- Analytical component: Star-Schema benchmark (SSB)
- Transactional component: adapted version of TPC-C benchmark

Practical and efficient ways to measure throughput frontier

Slightly modified schema to measure freshness

A Comparison Across HTAP Systems



There is still room for improving current HTAP systems

– Higher freshness for isolated designs

– Better performance isolation between T and A workloads

Outline

Demand of real-time analytics

HATtrick benchmark

- Throughput frontier
- Freshness

Hermes

- Off-the-shelf real-time analytics
- Transactional analytics
- Evaluation

Limitations of Current HTAP Solutions

Poor plug-and-play integration with existing TP and AP engines

Costly and time-consuming migrations

Not optimized for ***Transactional Analytics*** workloads

- Transactional and analytical logic execute in a single ACID transaction

Hermes: Off-the-Shelf Real-Time Analytics

Hermes aims to address current HTAP limitations

- A system layer between the existing engines and the storage
- Support for **pluggable engines**
- Efficient execution of **transactional analytics** with correct isolation level

Outline

Demand of real-time analytics

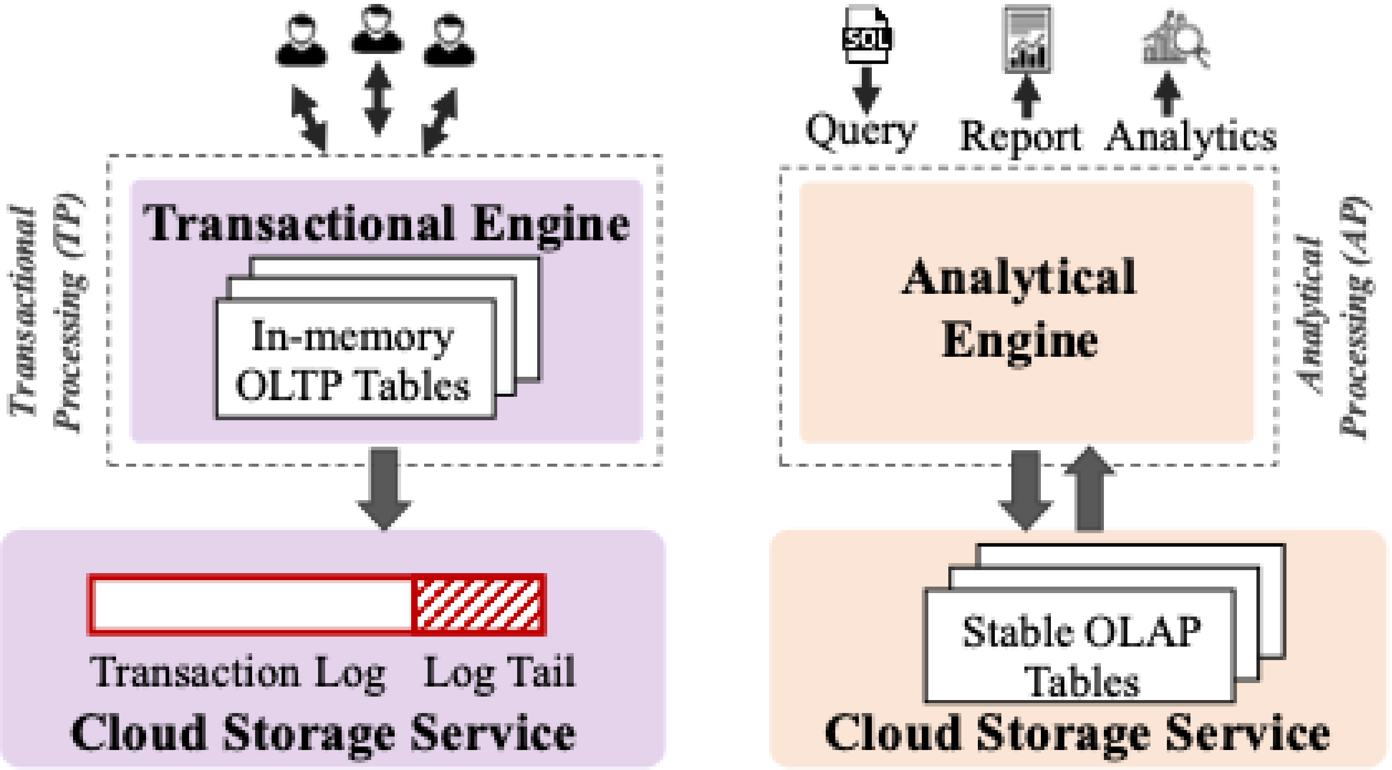
HATtrick benchmark

- Throughput frontier
- Freshness

Hermes

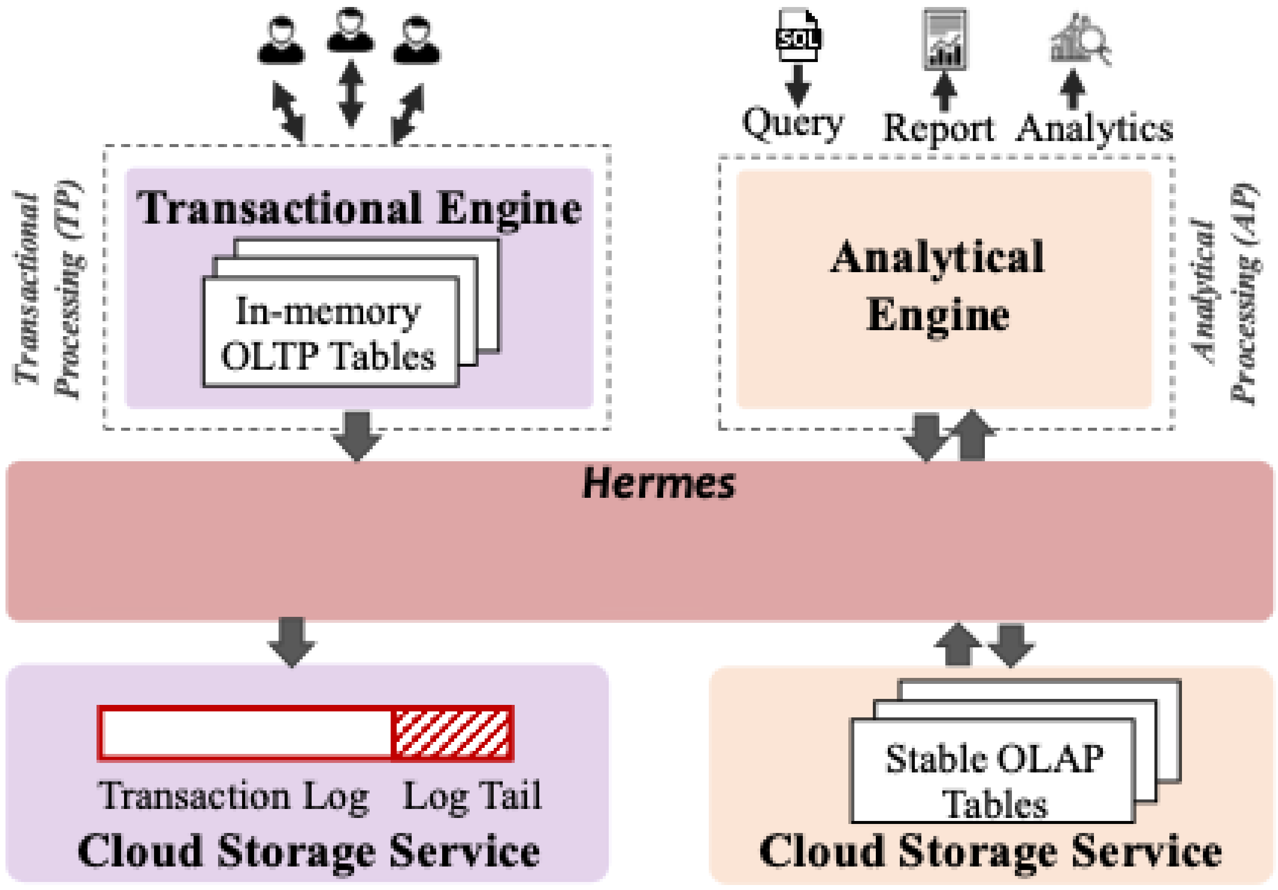
- **Off-the-shelf real-time analytics**
- Transactional analytics
- Evaluation

Separate TP/AP Engines



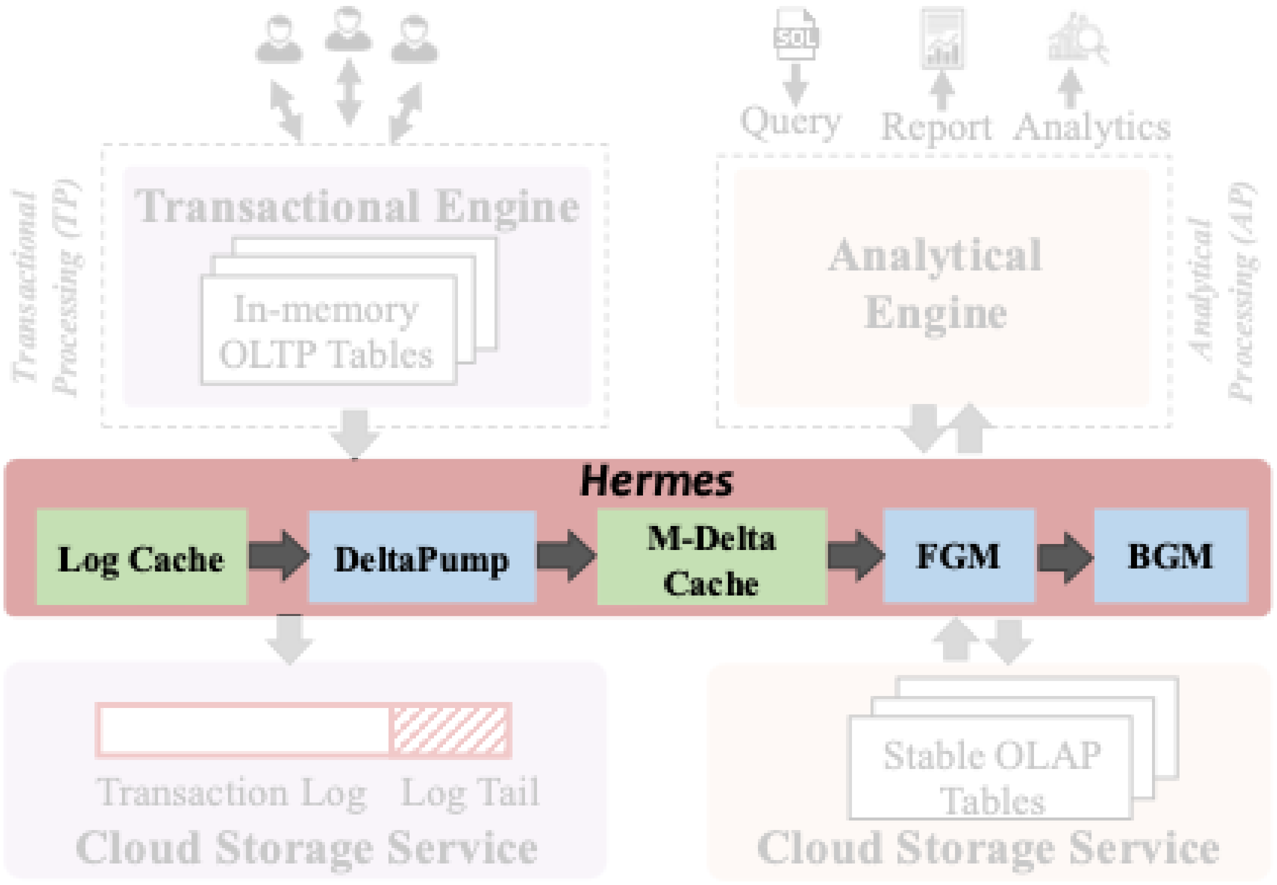
- TP engine handles transactions
- AP engine serves analytical queries
- Data storage is decoupled

Hermes Architecture



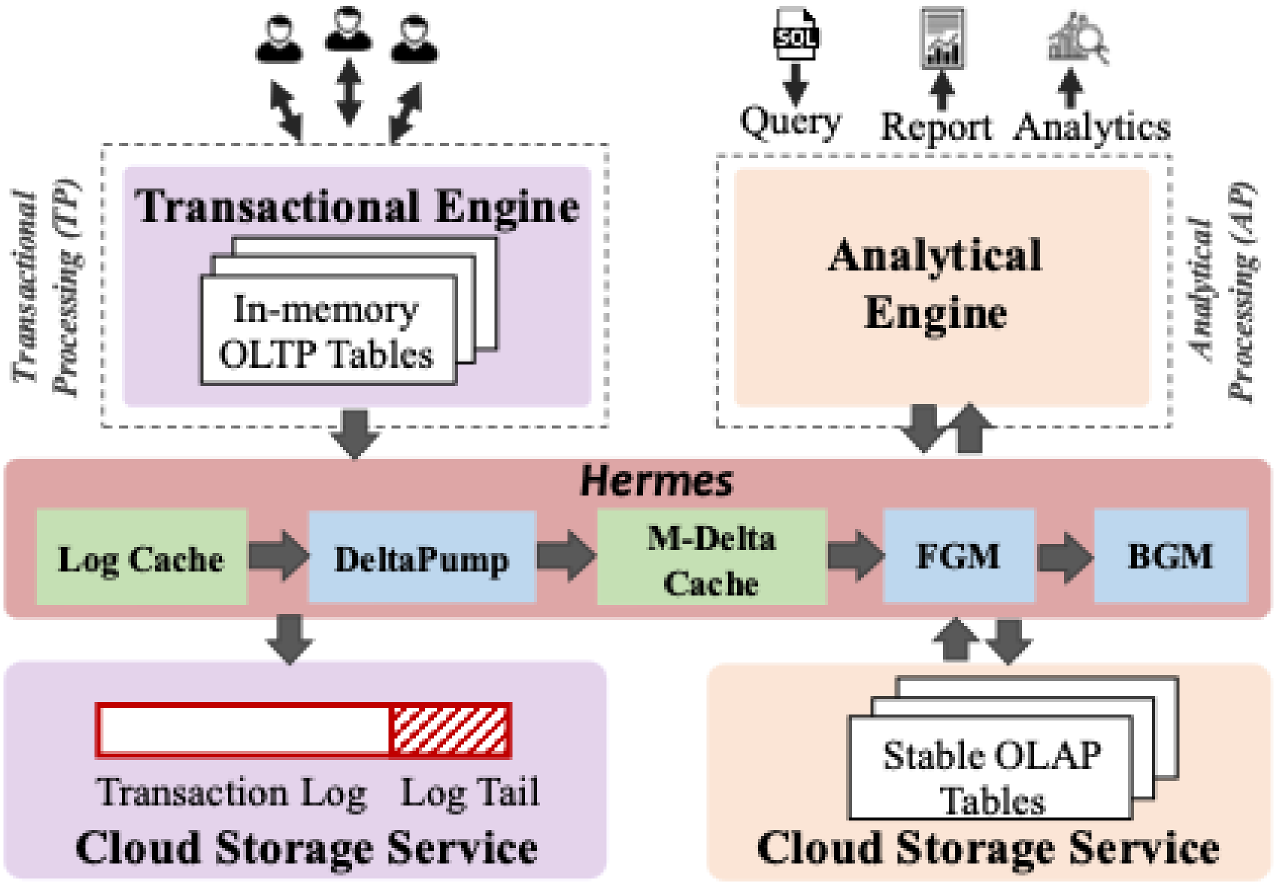
- A layer between engines and storage
- Leverages existing engines and storage
- **No changes** to existing engines

Hermes Architecture



Log Cache: in-memory log
DeltaPump: log parsing
Memory Delta Cache: in-memory parsed log
ForeGround Merge (FGM): merge of log with analytical reads
BackGround Merge (BGM): async merge of log into stable data

Hermes Architecture



Outline

Demand of real-time analytics

HATtrick benchmark

- Throughput frontier
- Freshness

Hermes

- Off-the-shelf real-time analytics
- **Transactional analytics**
- Evaluation

Transactional Analytics

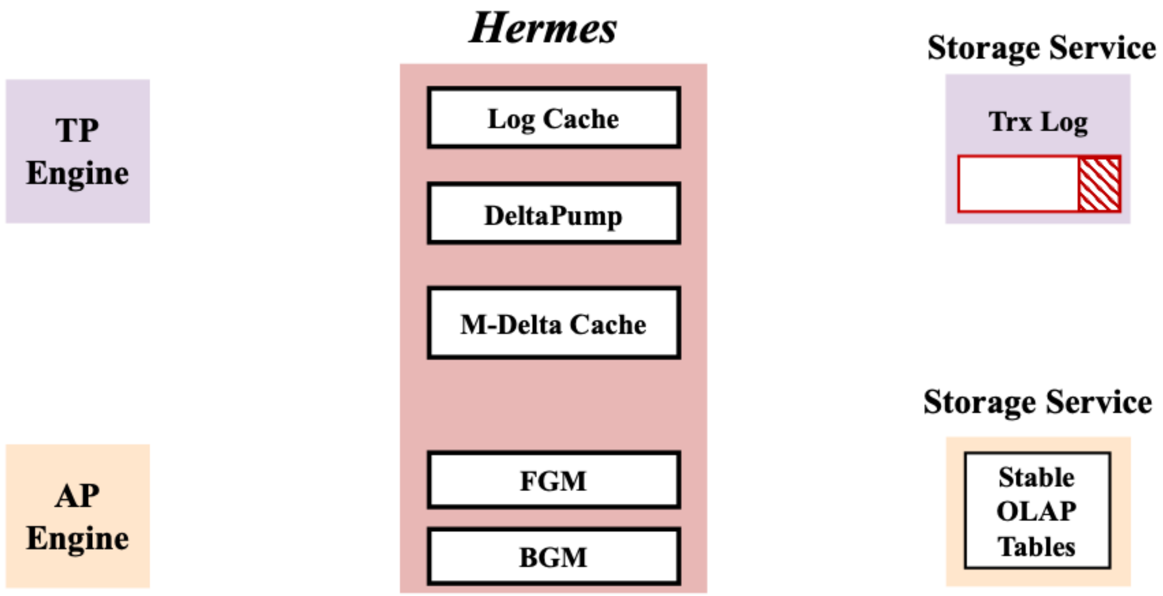
Integrate transactional and analytical operations into a single workflow

- Execute complex analytical queries within transactional context
- Enable immediate transactional updates based on analytical insights

Simplify application development and enable new application scenarios

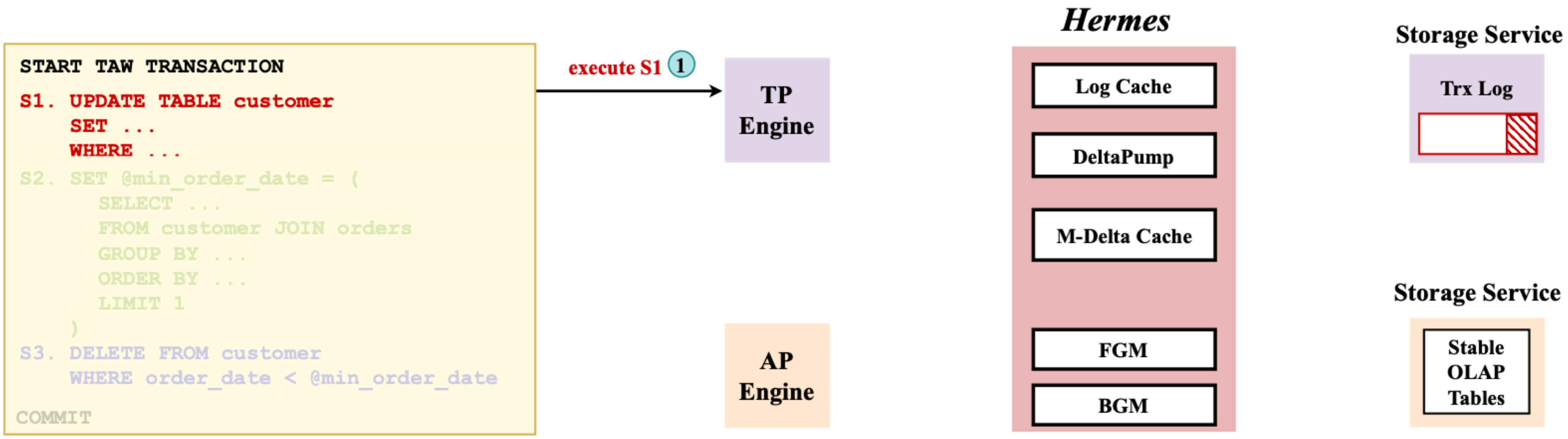
Transactional Analytics Example

```
START TAW TRANSACTION
S1. UPDATE TABLE customer
    SET ...
    WHERE ...
S2. SET @min_order_date = (
      SELECT ...
      FROM customer JOIN orders
      GROUP BY ...
      ORDER BY ...
      LIMIT 1
    )
S3. DELETE FROM customer
    WHERE order_date < @min_order_date
COMMIT
```



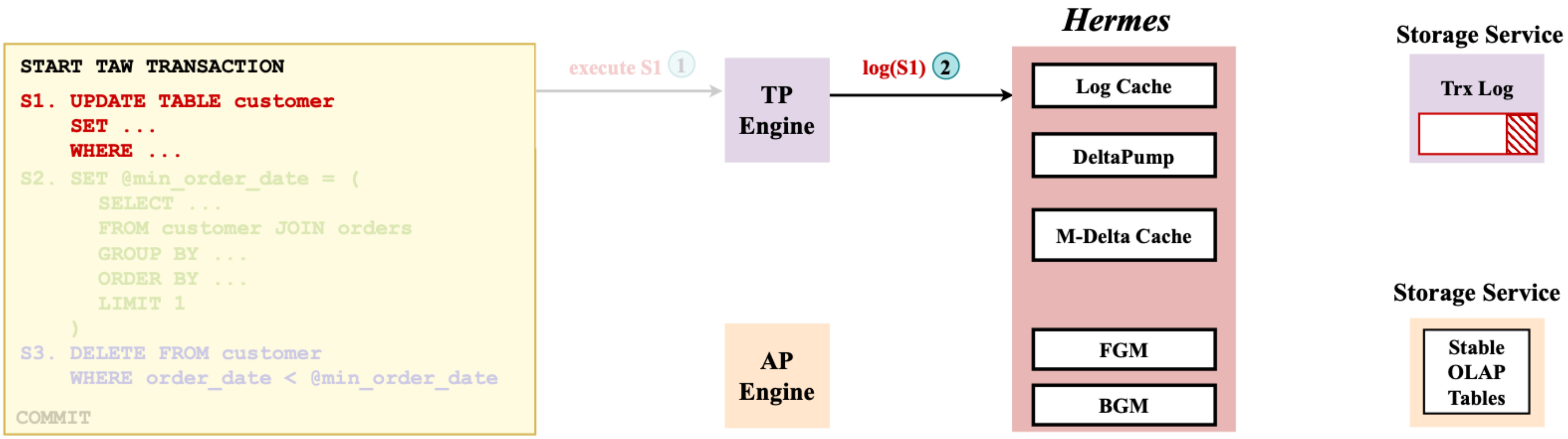
Transactional Analytics Example

- Insert, Update & Delete statements are directed to the TP engine



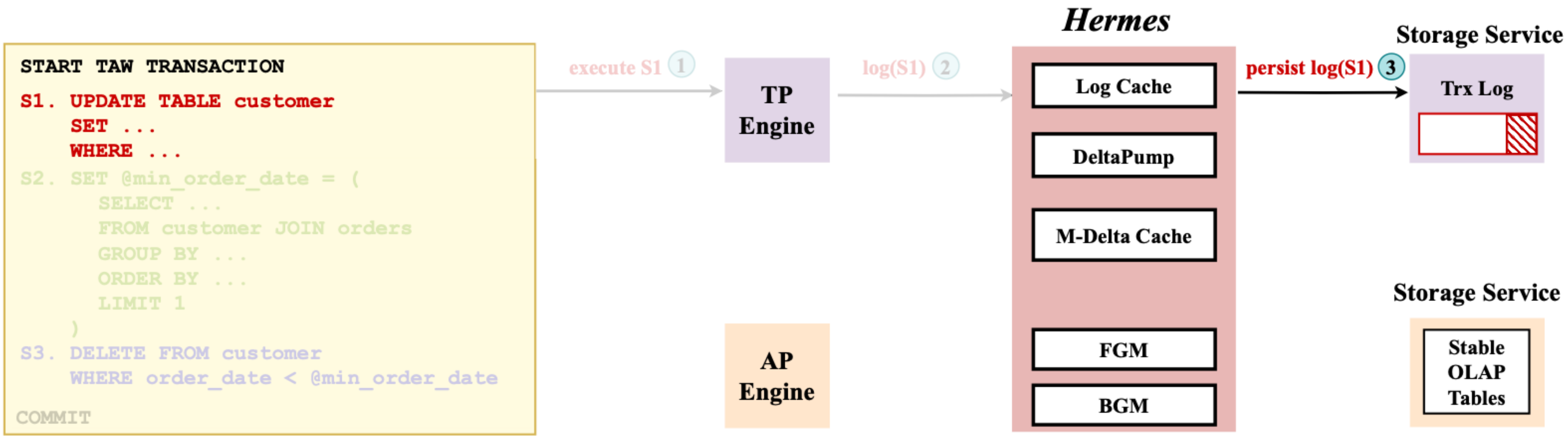
Transactional Analytics Example

- Log is both cached and durably persisted



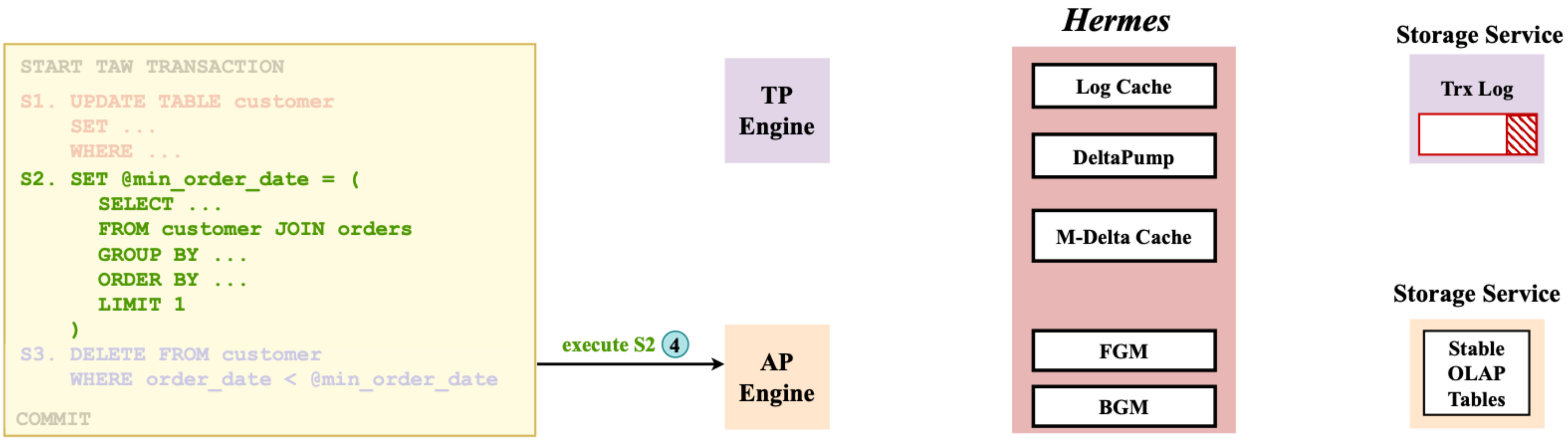
Transactional Analytics Example

- Log is both cached and durably persisted



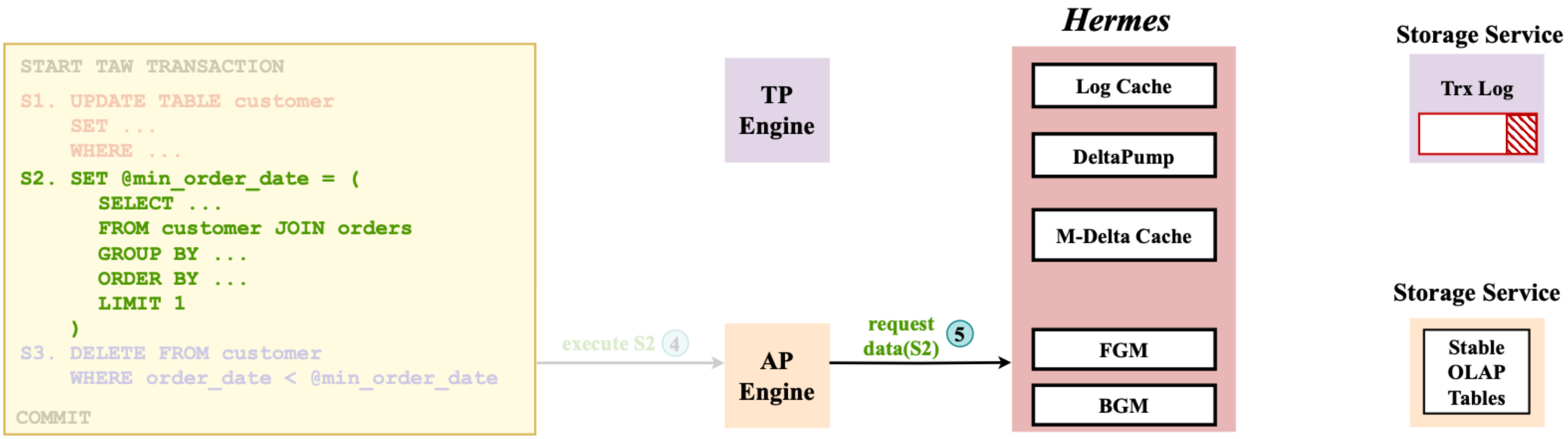
Transactional Analytics Example

- Analytical queries are directed to the AP engine



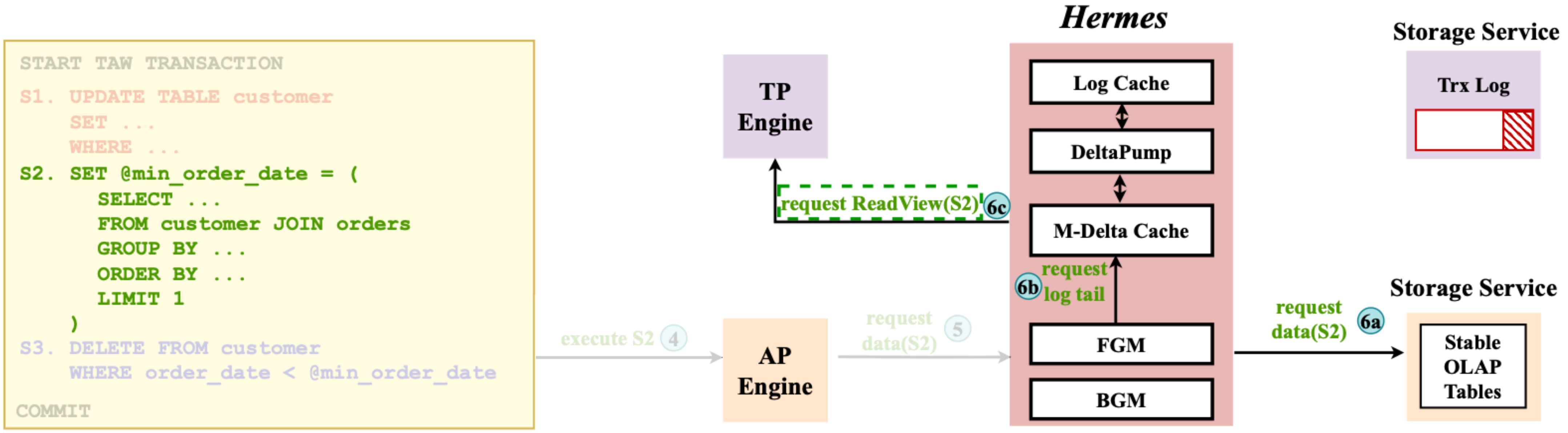
Transactional Analytics Example

- AP engine data requests are directed to *HERMES*



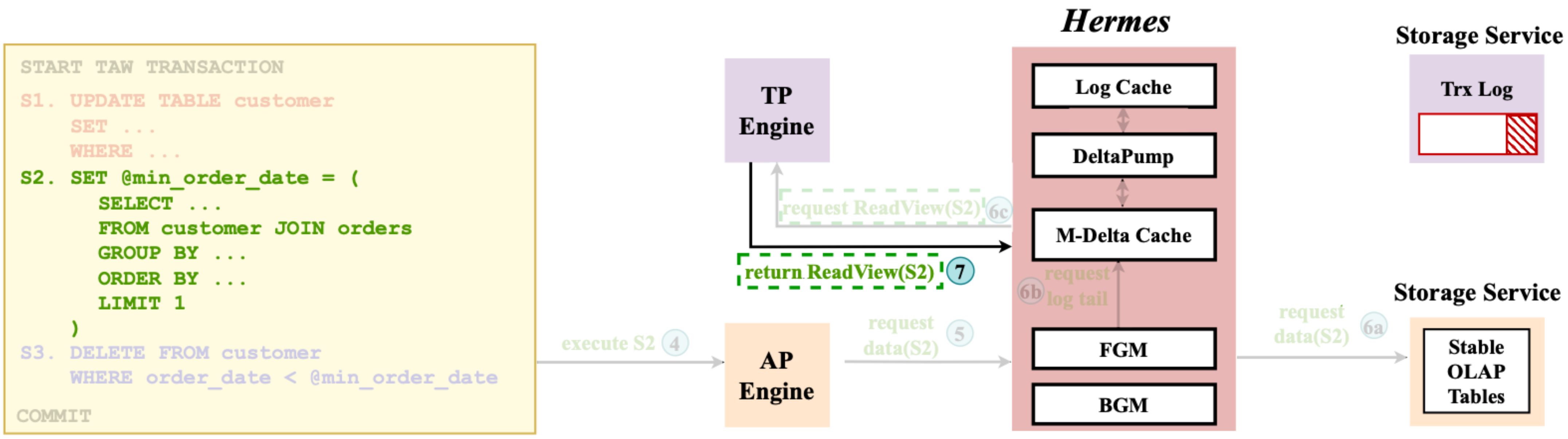
Transactional Analytics Example

- *HERMES* requests the ***ReadView*** *from* TP engine **(list of visible transactions)**
- In parallel, *HERMES* fetches log tail & stable data



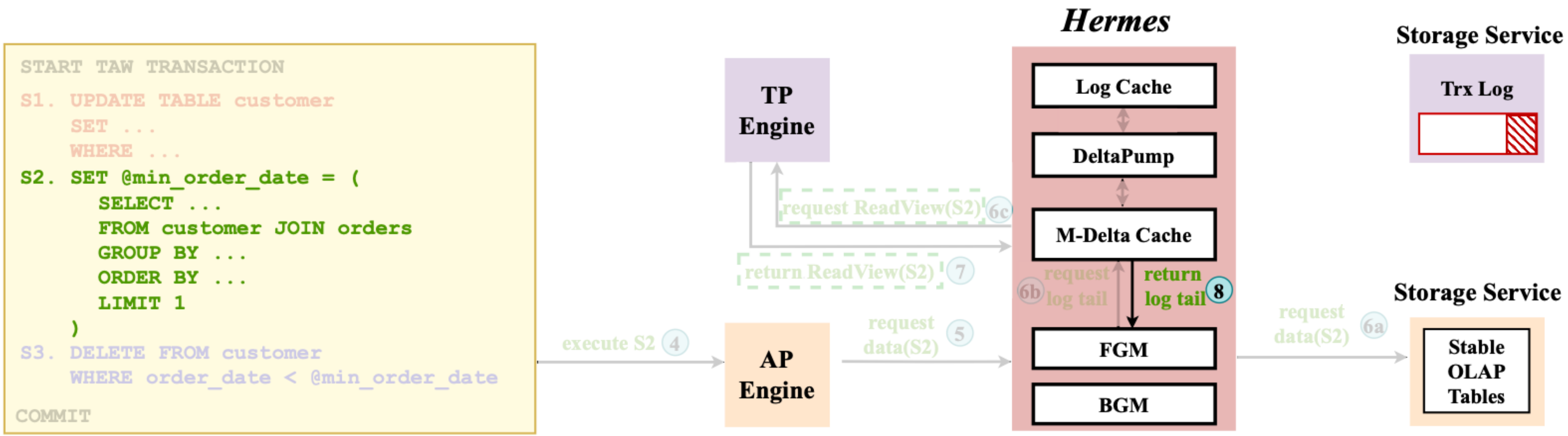
Transactional Analytics Example

- HERMES **waits for the ReadView** to arrive before initiating merging process



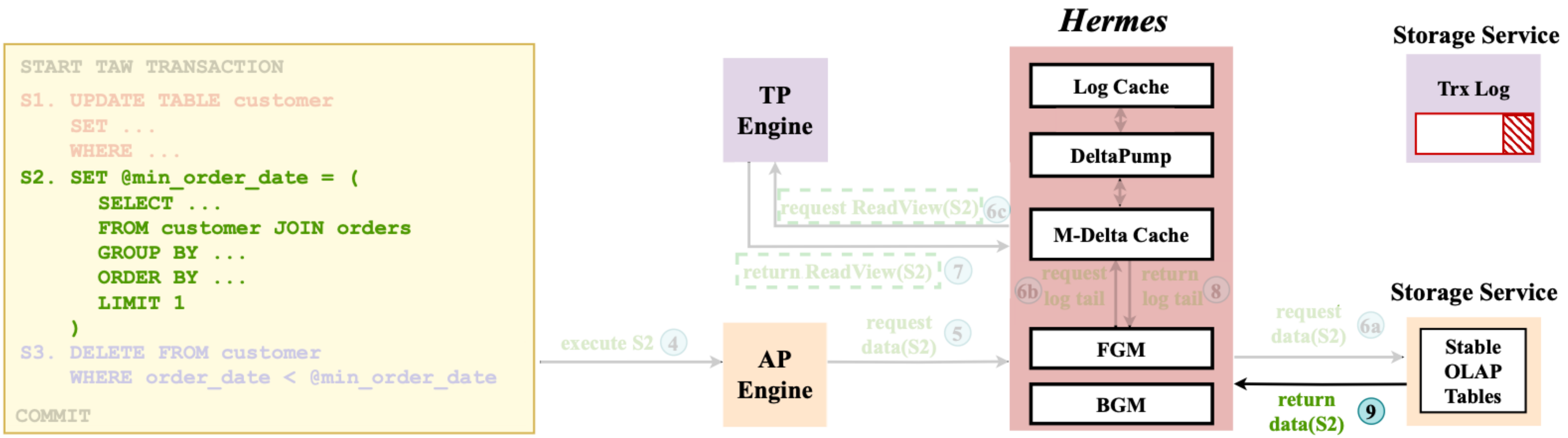
Transactional Analytics Example

- Using *ReadView HERMES* decides which log entries to merge with stable data



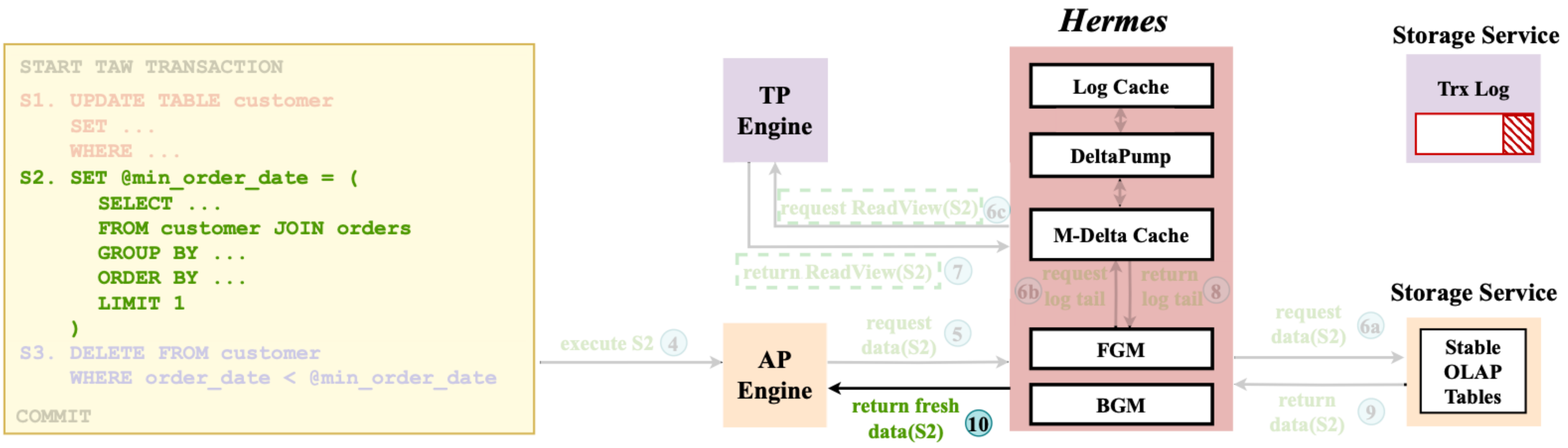
Transactional Analytics Example

- *HERMES* merges stable data with the correct data from the log



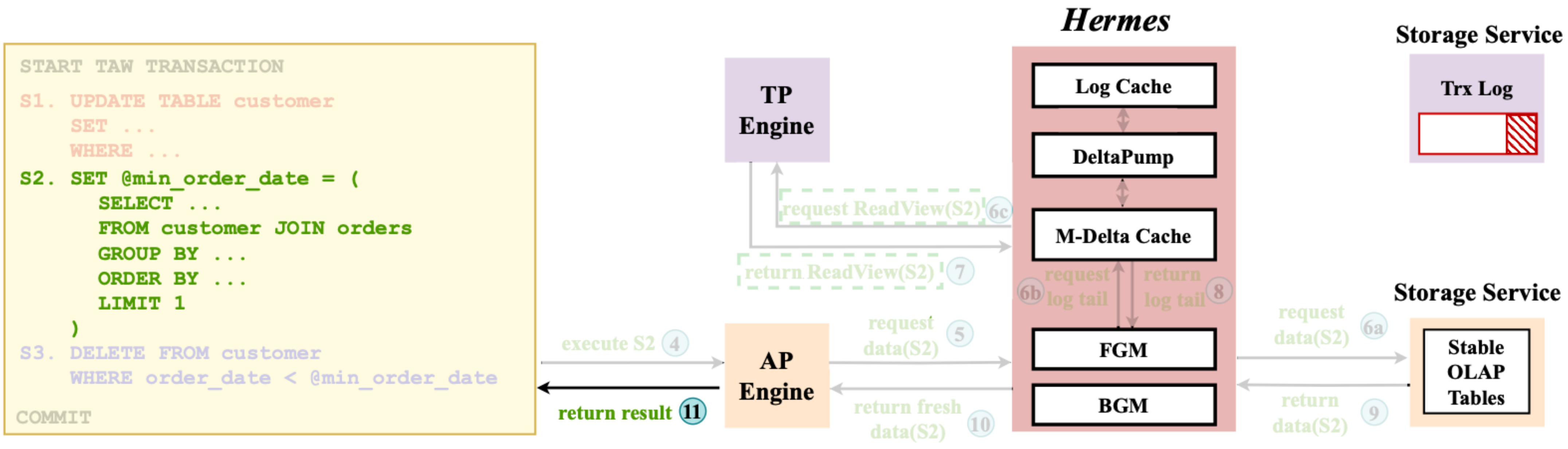
Transactional Analytics Example

- The **fresh stable data** are returned to the AP engine to continue the rest of query plan



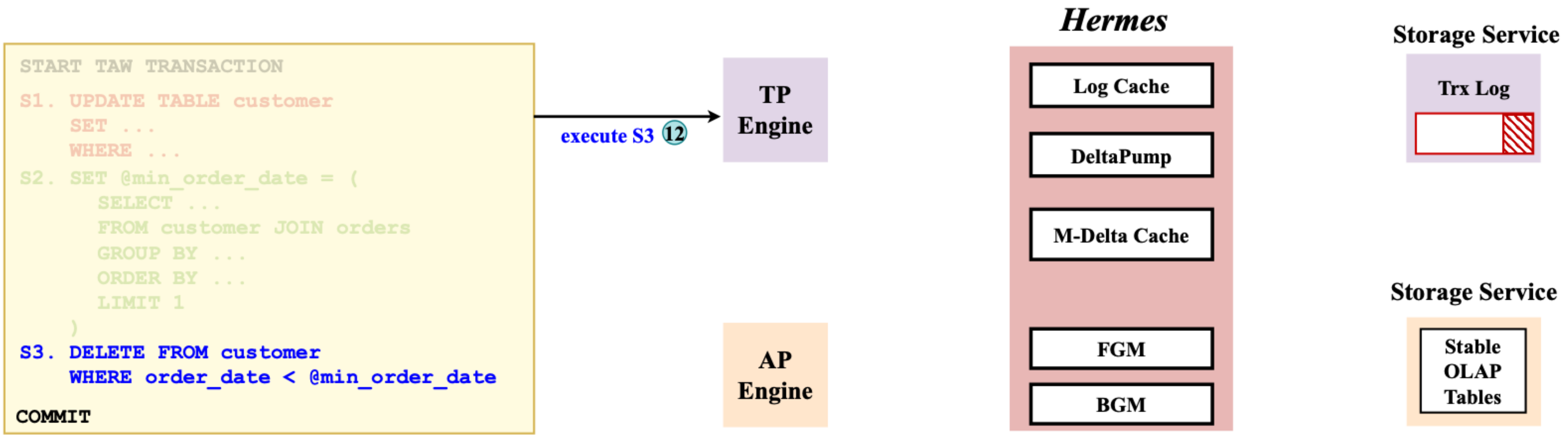
Transactional Analytics Example

- After execution results are returned to the application



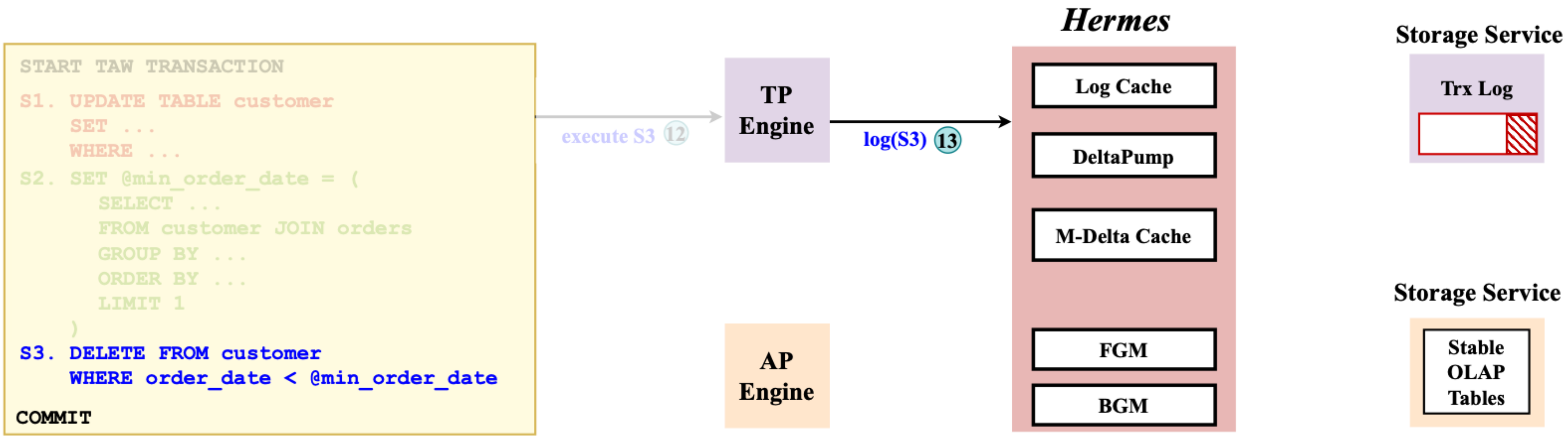
Transactional Analytics Example

- Insert, Update & Delete statements are directed to the TP engine



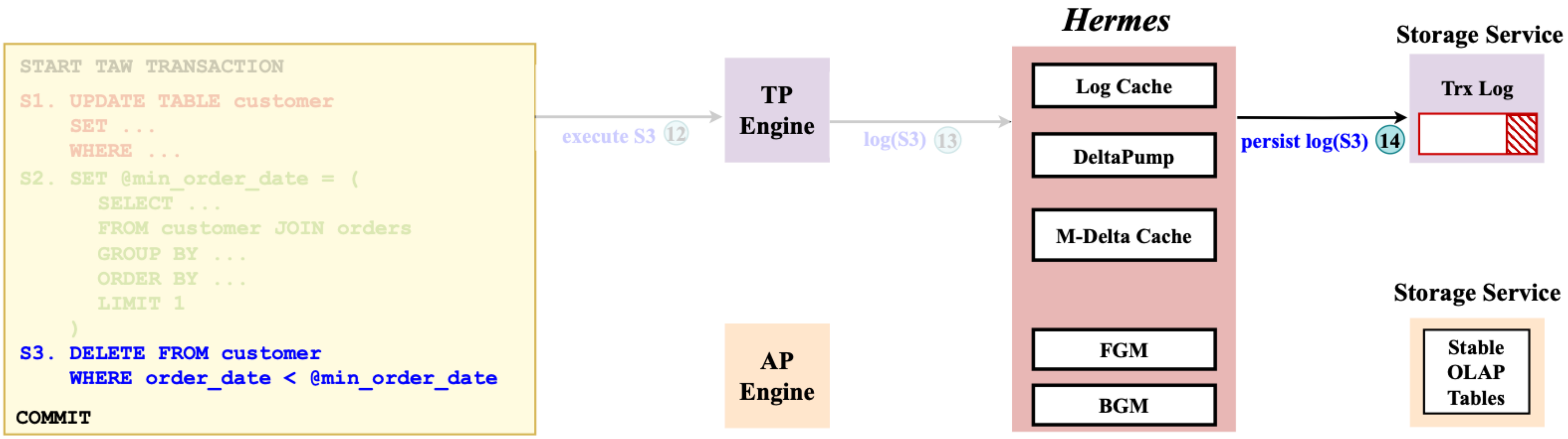
Transactional Analytics Example

- Log is both cached and durably persisted

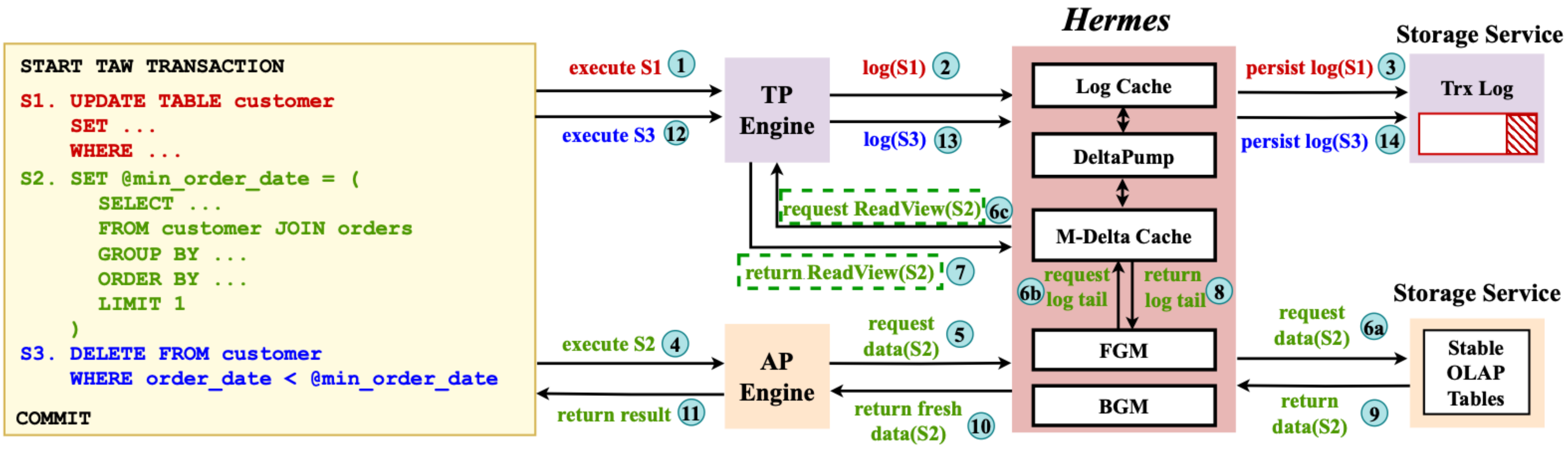


Transactional Analytics Example

- Log is both cached and durably persisted



Transactional Analytics Example



Outline

Demand of real-time analytics

HATtrick benchmark

- Throughput frontier
- Freshness

Hermes

- Off-the-shelf real-time analytics
- Transactional analytics
- **Evaluation**

Hermes Evaluation

Engine selection for *HERMES*:

- TP engine: MySQL
- AP engine: DuckDB

Workload: TAW: Transactional Analytics workload

Hardware setup:

- **HERMES layer:** c5.9xlarge, **MySQL:** c5.4xlarge, **DuckDB:** c5.4xlarge
- **TiDB:** 12 x c5.4xlarge (recommended configuration)
- **MySQL:** c5.12xlarge

TP- and AP-Only Workloads

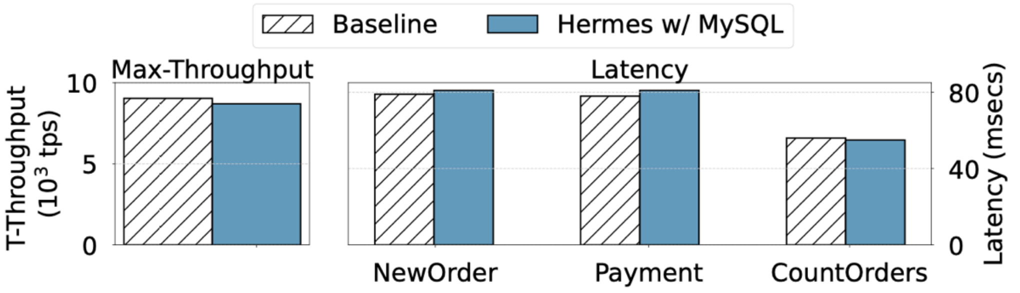
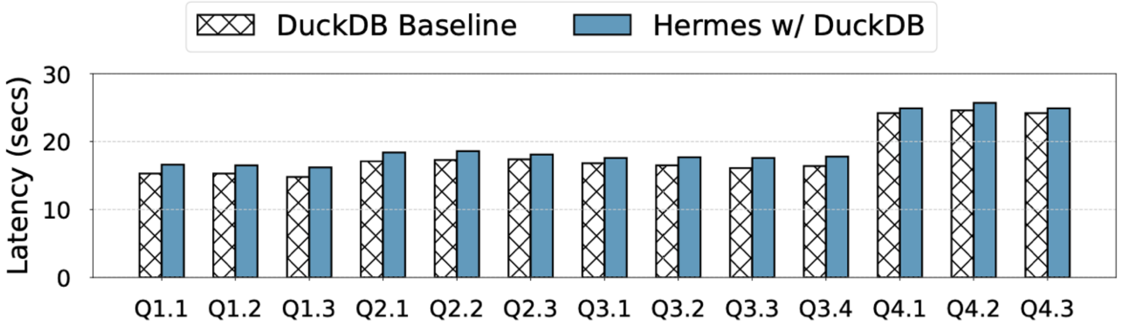


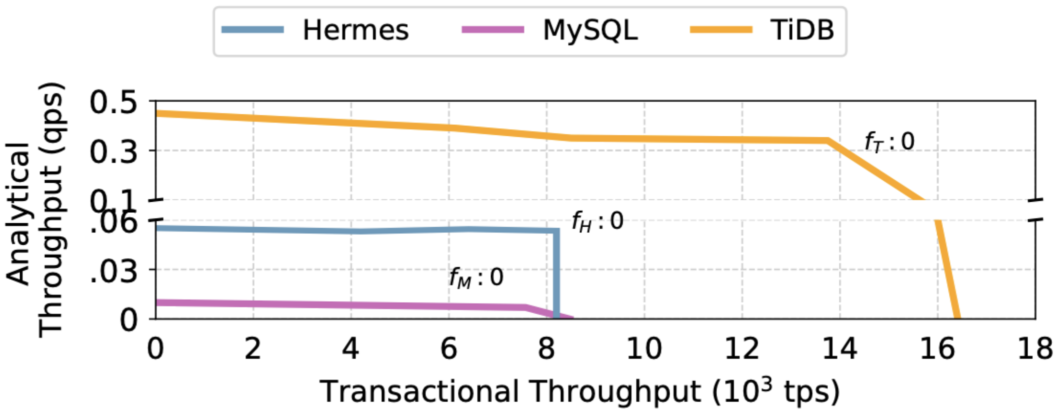
Figure 3: Transactional throughput (T-Throughput) results in tps (left) and transactions' latency results in msecs (right) when executing HATtrick in Hermes w/ MySQL vs. the standalone MySQL executing transactions.



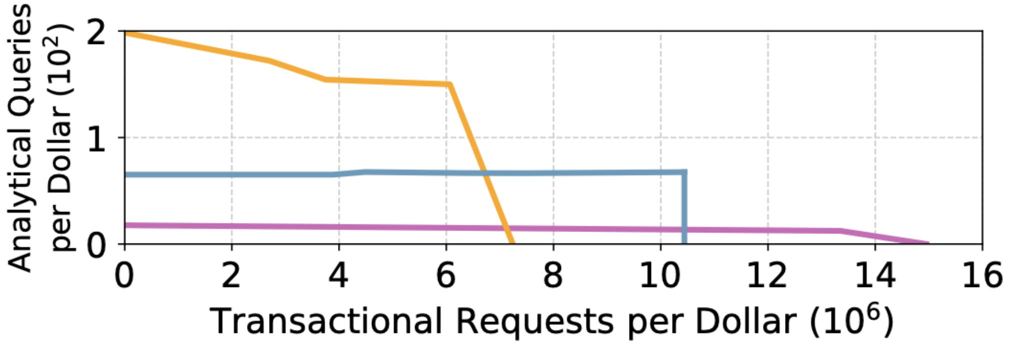
(b) DuckDB vs. Hermes w/ DuckDB.

Hermes incurs minimal overhead for TP- and AP-only workloads

HATtrick Benchmark



(a) Throughput frontiers and avg. freshness scores.

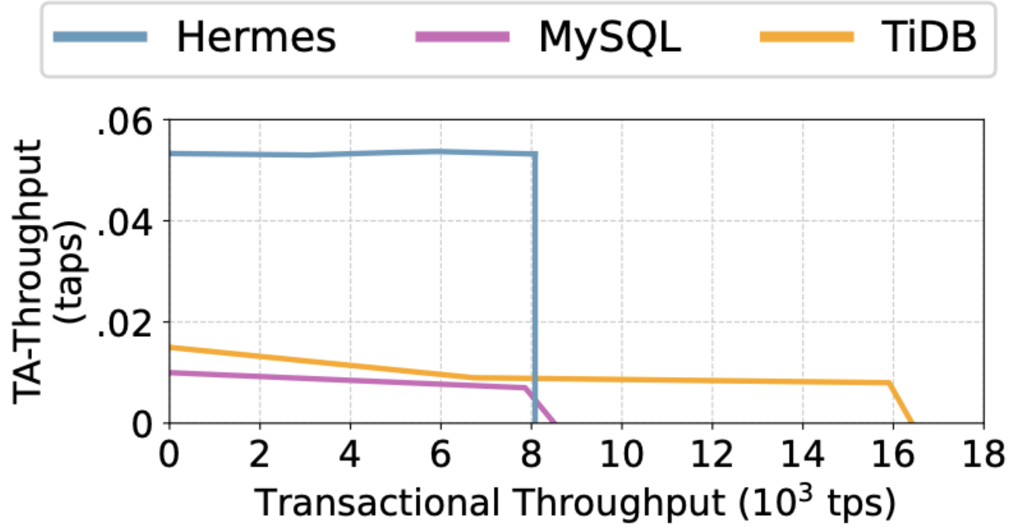


(b) Operations per dollar frontiers.

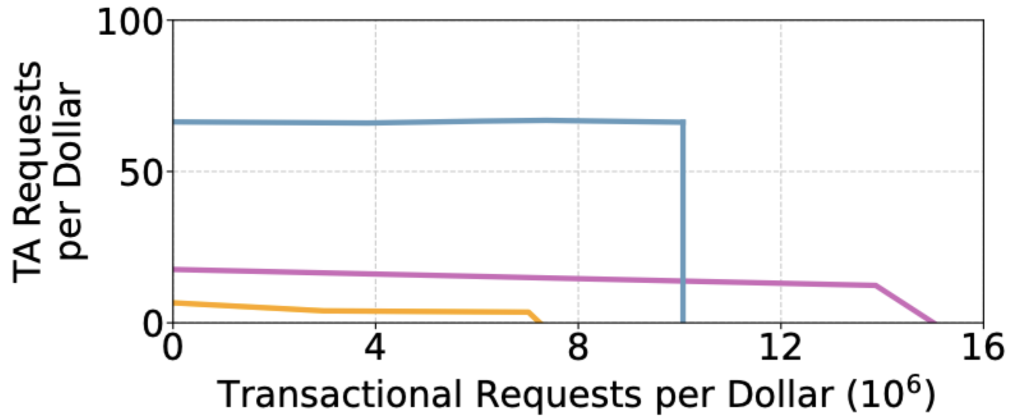
Figure 6: Throughput frontiers, freshness scores (f_{DB}) and operations per dollar frontiers for Hermes (with MySQL and DuckDB), MySQL, and TiDB when executing HATtrick.

- Hermes achieves near-perfect isolation between TP and AP workloads
- Hermes is competitive in terms of operations per dollar

TAW Benchmark



(c) Throughput frontiers in SI.



(d) Operations per dollar frontiers in SI.

TAW= Transactional Analytics workload

Hermes has great throughput frontier compared to baselines

HTAP – Q/A

Read consistency with multiple concurrent updates?

Very large log tails?

Schema changes or schema mismatches?

What about command logging?

What if Hermes layer fails? Recover log cache, M-Delta cache?

Hermes integrates multiple AP/TP systems simultaneously?

Next Lecture

Xinjing Zhou, et al., Two is Better Than One: The Case for 2-Tree for Skewed Data Sets. CIDR 2023