A Simple and Reliable Benchmark for Data Freshness in HTAP

Takaaki Kanetsuki[1,*,†], Toshiyuki Amagasa[1,†]

[1]*University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, Japan*

Abstract

In Hybrid Transactional/Analytical Processing (HTAP) systems, particularly those employing separate storage engines for OLTP and OLAP, data freshness—defined as the time lag between a transaction commit and its visibility in analytical queries—is a critical performance metric. While existing benchmarks like HyBench provide freshness metrics, they often lack portability, requiring specific datasets or intrusive schema modifications (e.g., adding timestamp columns to every business table), which limits their applicability to running industrial systems. To address this, we propose a simple and reliable method to evaluate data freshness without compromising the original database schema. Our approach involves adding a single auxiliary table to record periodic heartbeats and calculating the freshness gap under Snapshot Isolation or higher to ensure consistency. Experiments using PostgreSQL and TiDB demonstrate that our method accurately detects synchronization delays and exposes physical visibility lags in distributed environments that theoretical configurations often overlook.

Keywords

HTAP, Data freshness

1. Introduction

Hybrid Transactional/Analytical Processing (HTAP) systems [1, 2, 3, 4, 5, 6, 7] enable the integration of transactional (OLTP) and analytical (OLAP) workloads within a single platform, allowing for real-time analysis of operational data. A common architectural pattern in HTAP, particularly to isolate workload interference, is the separate storage model, where data is synchronized from a row-oriented OLTP store to a column-oriented OLAP store. In this architecture, a critical performance metric is data freshness, it indicates the time delay between a data modification in the transactional store and its visibility in the analytical store. In domains requiring immediate decision-making, such as FinTech or fraud detection, maintaining high data freshness is as critical as throughput or latency. However, accurately measuring data freshness in existing industrial applications poses significant challenges.

While standard benchmarks like CH-benCHmark [8], HATtrick [9] and HyBench [10] provide comprehensive methodologies for evaluating HTAP systems, they generally lack portability. These benchmarks require generating specific datasets and schemas to run their tests. Consequently, they cannot be easily applied to monitor the freshness of an already running production database without intrusive schema modifications. Furthermore, relying solely on the theoretical configuration of data synchronization is insufficient. In real-world high-load scenarios, resource contention, network jitter, or lock conflicts often cause the actual data freshness to deviate significantly from the configured synchronization delay. There is a lack of lightweight methods to verify this actual freshness from a user's perspective (black-box measurement) without disrupting the application logic.

To address these issues, we propose a simple, reliable, and portable method for evaluating data freshness. Unlike existing complex benchmarks, our approach requires only the addition of a single auxiliary table, leaving the original database schema untouched. Our method focuses on systems

DOLAP 2026: 28th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, co-located with EDBT/ICDT 2026, March 24, 2026, Tampere, Finland

*Corresponding author.

†These authors contributed equally.

takaaki@kde.cs.tsukuba.ac.jp (T. Kanetsuki); amagasa@cs.tsukuba.ac.jp (T. Amagasa)
0000-0003-0595-2230 (T. Amagasa)

running at Snapshot Isolation (SI) or higher, ensuring that freshness is measured consistently without the ambiguity of phantom reads The contributions of this paper are as follows.

1. We propose a lightweight measurement method that ensures high portability, allowing it to be *plugged in* to existing HTAP systems with minimal overhead.
2. We demonstrate the validity of this method through experiments on both PostgreSQL (monolithic) and TiDB (distributed), confirming its ability to detect freshness degradation caused by system delays.

2. Related work

Several benchmarks have been proposed to evaluate HTAP systems [11, 8, 12, 13]. The CH-benCHmark combines TPC-C (OLTP) and TPC-H (OLAP) workloads to assess mixed-workload performance. HyBench improves upon this by simulating more realistic transaction-analysis interactions and defines four key principles for HTAP benchmarks: *relevance*, *portability*, *scalability*, and *simplicity* (described in Hybench paper [10]).

While these benchmarks are excellent for comparing different HTAP systems (e.g., during technology selection), they are less suitable for measuring data freshness in existing production environments. This is because they require specific schemas and datasets to run. Furthermore, HyBench requires adding a dedicated `FRESH_TS` column to every table to track freshness. For a running industrial application, modifying the schema of core tables solely for monitoring purposes introduces unacceptable maintenance overhead and risk. Our approach differs by focusing on portability, we aim to measure freshness in any schema by adding only a single auxiliary table, treating the application logic as a black box.

Data freshness in HTAP is often defined as *the time lag between a transaction commit and its visibility in analytical queries* [2]. However, the measurement of this lag depends heavily on the Transaction Isolation Level. While critical for ensuring data consistency, this aspect has not been widely considered in the context of HTAP benchmarking. In database systems, isolation levels control the visibility of uncommitted or newly committed data. Under Read Uncommitted, analytical queries might read dirty data, making *freshness* ambiguous. To ensure reliable analytics, modern HTAP systems often employ SI or Serializable levels. Under SI, a query sees a consistent snapshot of the database as of the query's start time. Therefore, accurately measuring freshness requires a mechanism that respects these snapshot boundaries—ensuring that the measured *freshness* corresponds to a valid snapshot timestamp, preventing phantom reads. Our work specifically targets systems providing SI or higher to guarantee this consistency.

3. Proposed Approach

We propose a lightweight, portable method to evaluate *data freshness* in HTAP systems. Our design philosophy prioritizes portability (requiring no changes to existing application schemas) and observability (verifying actual data visibility rather than theoretical configuration).

3.1. Target Architecture and Isolation Requirement

We target the Separate Storage HTAP architecture (Figure 1), where OLTP (row-store) and OLAP (column-store) engines are decoupled and synchronized via mechanisms like Change Data Capture (CDC) or ETL pipelines.

- **The Necessity of Measurement** A common critique is that if the data synchronization interval (e.g., ETL frequency) is known, freshness is implicitly known. We argue this assumption holds only in ideal conditions. In high-load industrial environments, *silent delays* frequently occur due to network congestion, lock contention, or resource starvation (CPU/IO) on the OLAP side.

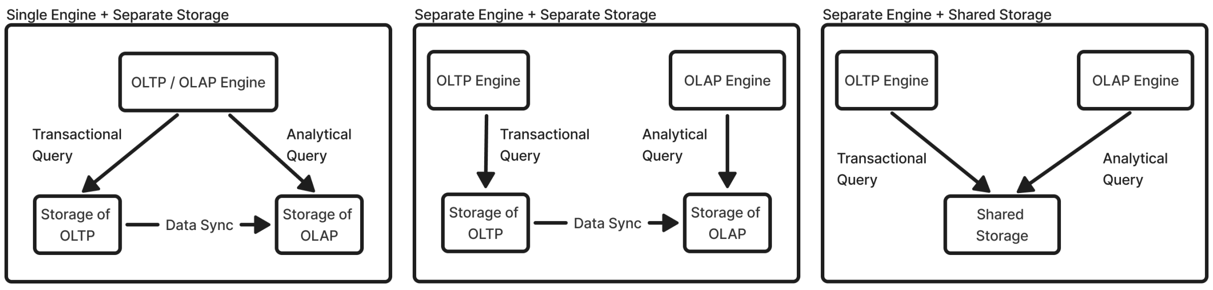


Figure 1: Target HTAP architectures.

Table 1
HTAP data synchronization.

Architecture	Data sync
Single engine + separate storage	Yes
Separate engine + separate storage	Yes
Separate engine + shared storage	No

Therefore, a black-box measurement approach is essential to validate whether the configured freshness matches the actual visibility.

- **Isolation Level** To ensure the measured timestamp corresponds to a consistent snapshot of the database, the target system must support SI or higher. Lower isolation levels (e.g., Read Committed) may introduce ambiguity due to phantom reads during the timestamp retrieval.

3.2. Measurement Workflow

The proposed method involves three steps, as illustrated in Figure 2. We introduce a single auxiliary table, `data_freshness_info`, to both the OLTP and OLAP stores. Unlike HyBench, which requires modifying every existing table, our method leaves the business schema untouched.

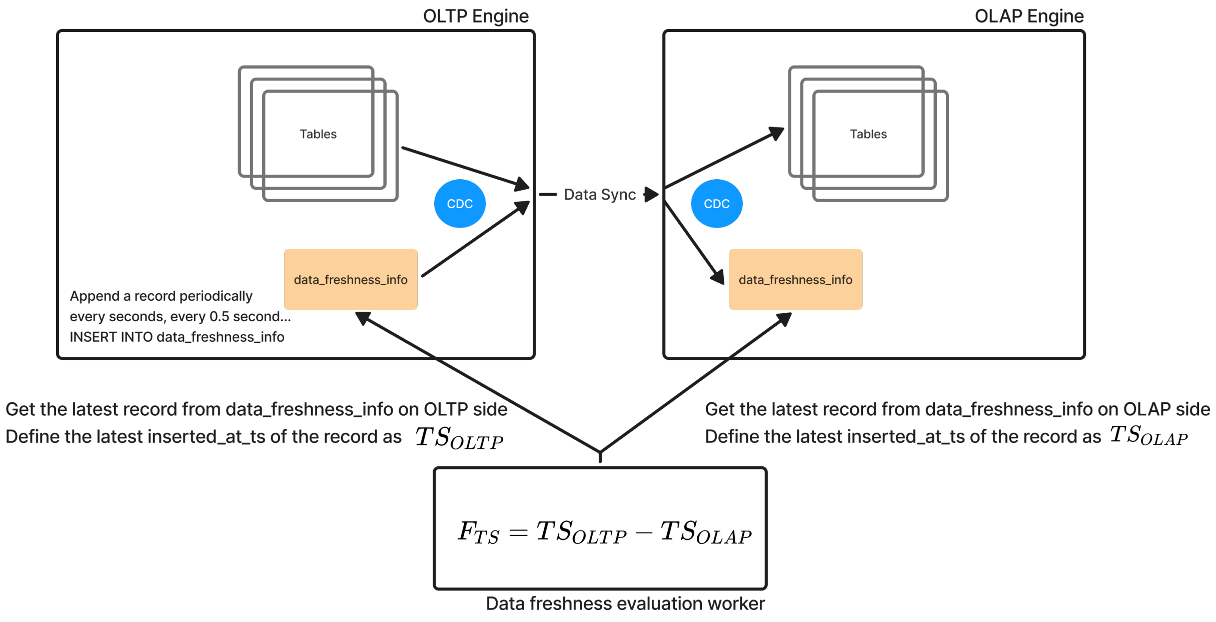

OLTP Engine
Tables
CDC
data_freshness_info
Append a record periodically
every seconds, every 0.5 second...
INSERT INTO data_freshness_info
Data Sync
OLAP Engine
Tables
CDC
data_freshness_info
Get the latest record from data_freshness_info on OLTP side
Define the latest inserted_at_ts of the record as TS_{OLTP}
Get the latest record from data_freshness_info on OLAP side
Define the latest inserted_at_ts of the record as TS_{OLAP}
$F_{TS} = TS_{OLTP} - TS_{OLAP}$
Data freshness evaluation worker

Figure 2: Data sync.

Heartbeat Generation (OLTP Side) A dedicated worker periodically inserts the current timestamp into the OLTP side's `data_freshness_info` table (e.g., every 0.5 seconds). This stream of updates mimics a standard transactional workload and propagates to the OLAP side via the system's synchronization mechanism.

Freshness Calculation (OLAP Side) To measure freshness F_{TS}, a monitor worker queries both the OLTP and OLAP sides to retrieve the latest visible timestamp.

- TS_{OLTP} The latest timestamp committed to the OLTP primary.
- TS_{OLAP} The latest timestamp visible in the OLAP replica.
- Data freshness is then calculated as F_{TS} = TS_{OLTP} - TS_{OLAP}

3.3. Implementation Considerations

Since the monitoring worker continuously inserts timestamp records into the data_freshness_info table, the table size will naturally grow over time. While modern database storage is abundant, unlimited growth is inefficient and consumes unnecessary storage resources. To mitigate this, we recommend implementing a housekeeping process: a scheduled background job that periodically deletes records exceeding a specific retention period (e.g., keeping only the last hour of data).

Critically, the performance of the freshness measurement does not degrade as the table grows. The measurement query employs a ORDER BY id DESC LIMIT 1 clause, which allows the query optimizer to utilize the Primary Key index. The database engine only needs to access the tail of the index to retrieve the latest record, avoiding a full table scan. Therefore, the query execution time remains minimal and constant (O(1) effective access time) regardless of the total number of historical records in the table. This ensures that the monitoring mechanism itself does not become a bottleneck, even if the housekeeping interval is relaxed.

4. Experiments

We conducted experiments to verify whether the proposed method can accurately measure data freshness under enforced delay conditions. We utilized Docker, a containerized environment to ensure reproducibility, a key requirement for reliable benchmarking.

4.1. Experimental Setup Environment

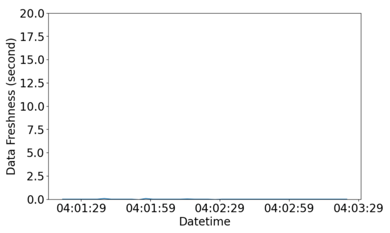
The experiments were performed on a machine equipped with an Apple M2 processor (8 cores) and 16GB RAM. We used PostgreSQL 14 (representing a monolithic architecture) and TiDB Serverless (representing a distributed cloud-native HTAP architecture).

To simulate a realistic business workload, we designed a schema mimicking an e-commerce platform consisting of four tables (`office`, `office_member`, `stock`, `stock_manager`). We executed a mixed workload where an OLTP worker concurrently inserted/updated records (5 transactions/sec) to generate CDC traffic, while the monitoring worker inserted timestamps into the data_freshness_info table at 1-second intervals. We artificially introduced data synchronization delays and observed whether our method could detect them.

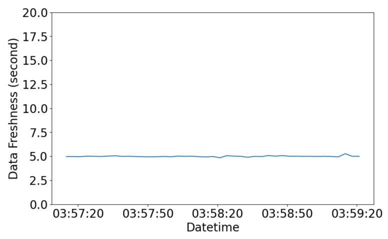
- **PostgreSQL** We utilized Streaming Replication and configured recovery_min_apply_delay to 0s, 5s, and 10s.
- **TiDB** We utilized the tidb_read_staleness session variable to enforce a read consistency lag of 0s, 5s, and 10s.

4.2. Results on PostgreSQL (Monolithic)

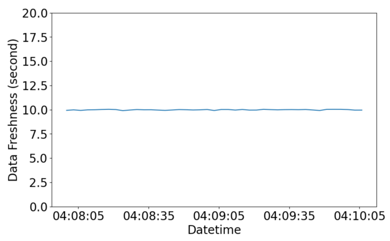
Figure 3 (a-c) illustrates the measured data freshness over time.



(a) delay=0s



(b) delay=5s



(c) delay=10s

Figure 3: Experimental results for different delay settings.

- **0s Delay** The freshness remained near 0 seconds, confirming minimal overhead in the replication mechanism.
- **5s and 10s Delay** The measured F_TS consistently stayed above the configured thresholds (5s and 10s, respectively)

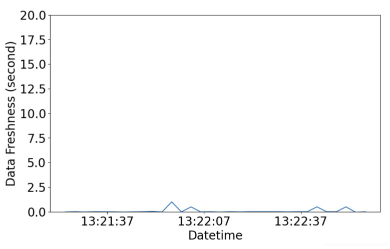
In the monolithic environment, the physical replication lag closely matches the logical configuration. Our tool accurately captured this behavior with high stability, validating the baseline correctness of the proposed method.

4.3. Results on TiDB (Distributed)

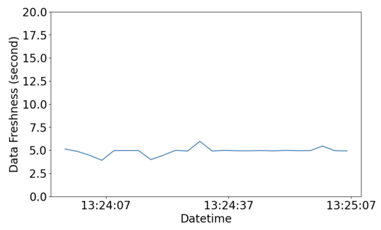
Figure 4 (a-c) presents the results for the distributed HTAP environment. Similar to PostgreSQL, the measured freshness generally followed the configured `tidb_read_staleness` settings (0s, 5s, 10s). Unlike the flat lines observed in PostgreSQL, the TiDB results exhibit minor fluctuations around the target values. These fluctuations are not measurement errors but reflect the inherent characteristics of distributed systems.

1. **Clock Synchronization (TSO)** TiDB uses a central Timestamp Oracle (TSO) for global consistency. Slight network latencies in fetching TSOs across distributed nodes (TiKV and TiFlash) can introduce micro-deviations when comparing timestamps between the client and the storage engine.
2. **Network Jitter** Since TiDB Serverless separates compute and storage over the network, communication jitter affects the instant visibility of updates.

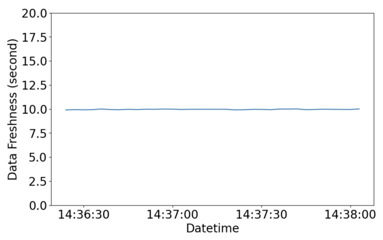
The results demonstrate the sensitivity of our tool. It reveals that in a distributed cloud environment, *configured consistency* does not always equal *physical real-time visibility*. Our method successfully exposes this reality gap, encompassing network and architectural jitters that theoretical calculations often overlook.



(a) staleness=0s



(b) staleness=5s



(c) staleness=10s

Figure 4: Experimental results with different tidb_read_staleness settings.

5. Conclusion

In this paper, we proposed a simple, reliable, and portable method for evaluating data freshness in HTAP systems. While existing benchmarks like HyBench provide comprehensive performance metrics, they often require intrusive schema modifications, making them unsuitable for monitoring running industrial applications. Our approach addresses this gap by introducing a lightweight black-box measurement technique—adding a single auxiliary table—that can be applied to any existing database schema. We implemented and validated our method on both monolithic (PostgreSQL) and distributed (TiDB) HTAP architectures.

- **Validity** The experiments confirmed that our tool accurately detects delays in data synchronization, distinguishing between healthy states and induced lags.
- **Observability** Crucially, our experiments on TiDB revealed that *configured consistency* (e.g., theoretical staleness settings) does not always match *physical real-time visibility* due to network jitter and distributed clock synchronization issues. Our tool successfully exposes this reality gap, providing engineers with a trustworthy metric for actual data availability.

5.1. Future Work

While the proposed method serves as a practical baseline, several challenges remain for future research.

1. **Support for Lower Isolation Levels** Currently, our method requires SI or higher to guarantee consistent timestamp retrieval without phantom reads. However, many legacy systems operate under Read Committed. We plan to investigate statistical approaches to estimate data freshness reliability in environments with relaxed isolation guarantees.
2. **Precision in Distributed Environments** As observed in the TiDB experiments, distributed systems introduce measurement noise due to TSO (Timestamp Oracle) access latency and clock skew. Future iterations of this tool will incorporate outlier detection algorithms and advanced clock synchronization logic to smooth out network-induced fluctuations, ensuring higher precision in cloud-native environments.

Acknowledgements

This paper is based on results obtained from the project, "Research and Development Project of the Enhanced infrastructures for Post-5G Information and Communication Systems" (JPNP20017), commissioned by the New Energy and Industrial Technology Development Organization (NEDO), JST CREST Grant Number JPMJCR22M2, and JSPS KAKENHI Grant Number JP23K24949.

Declaration on Generative AI

During the preparation of this work, the author used Gemini 3 Pro, GPT-5.1 and GPT-5.2 in order to: Translate and check the grammar of the entire manuscript. After using these tools/services, the author reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] M. Pezzini, D. Feinberg, N. Rayner, R. Edjlali, Hybrid transaction/analytical processing will foster opportunities for dramatic business innovation, Gartner, 2014. Pages 4–20.

[2] J. Chen, Y. Ding, Y. Liu, F. Li, L. Zhang, M. Zhang, K. Wei, L. Cao, D. Zou, Y. Liu, et al., Bytehtap: bytedance's htap system with high data freshness and strong data consistency, Proceedings of the VLDB Endowment 15 (2022) 3411–3424.

[3] D. Huang, Q. Liu, Q. Cui, Z. Fang, X. Ma, F. Xu, L. Shen, L. Tang, Y. Zhou, M. Huang, et al., Tidb: a raft-based htap database, Proceedings of the VLDB Endowment 13 (2020) 3072–3084.

[4] P.-Å. Larson, A. Birka, E. N. Hanson, W. Huang, M. Nowakiewicz, V. Papadimos, Real-time analytical processing with sql server, Proceedings of the VLDB Endowment 8 (2015) 1740–1751.

[5] H. Avni, A. Aliev, O. Amor, A. Avitzur, I. Bronshtein, E. Ginot, S. Goikhman, E. Levy, I. Levy, F. Lu, et al., Industrial-strength oltp using main memory and many cores, Proceedings of the VLDB Endowment 13 (2020) 3099–3111.

[6] D. Makreshanski, J. Giceva, C. Barthels, G. Alonso, Batchdb: Efficient isolated execution of hybrid oltp+ olap workloads for interactive applications, in: Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 37–50.

[7] J. Wang, T. Li, H. Song, X. Yang, W. Zhou, F. Li, B. Yan, Q. Wu, Y. Liang, C. Ying, et al., Polardb-imci: A cloud-native htap database system at alibaba, Proceedings of the ACM on Management of Data 1 (2023) 1–25.

[8] Database Group, Technical University of Munich, Ch-benchmark: Towards a hybrid workload for databases, 2011. URL: https://db.in.tum.de/research/projects/CHbenCHmark/index.shtml, accessed: 2024-10.

[9] E. Milkai, Y. Chronis, K. P. Gaffney, Z. Guo, J. M. Patel, X. Yu, How good is my htap system?, in: Proceedings of the 2022 International Conference on Management of Data, 2022, pp. 1810–1824.

[10] C. Zhang, G. Li, T. Lv, Hybench: A new benchmark for htap databases, Proceedings of the VLDB Endowment 17 (2024) 939–951.

[11] R. Cole, F. Funke, L. Giakoumakis, W. Guy, A. Kemper, S. Krompass, H. Kuno, R. Nambiar, T. Neumann, M. Poess, et al., The mixed workload ch-benchmark, in: Proceedings of the Fourth International Workshop on Testing Database Systems, 2011, pp. 1–6.

[12] Transaction Processing Performance Council, Tpc-c benchmark, 1992. URL: https://www.tpc.org/tpcc/, accessed: 2024-10.

[13] Transaction Processing Performance Council, Tpc-h benchmark, 1999. URL: https://www.tpc.org/tpch/, accessed: 2024-10.