BenchCouncil Transactions on Benchmarks, Standards and Evaluations 3 (2023) 100122

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Full length article

Benchmarking HTAP databases for performance isolation and real-time analytics

Guoxin Kang *, Simin Chen, Hongxiao Li

Institute of Computing Technology Chinese Academy of Sciences, China
University of Chinese Academy of Sciences, China

ARTICLE INFO

Keywords:
HTAP databases
Benchmark
Performance isolation
Real-time analytics

ABSTRACT

Hybrid **T**ransactional/**A**nalytical **P**rocessing (HTAP) databases are designed to execute real-time analytics and provide performance isolation for online transactions and analytical queries. Real-time analytics emphasize analyzing the fresh data generated by online transactions. And performance isolation depicts the performance interference between concurrently executing online transactions and analytical queries. However, HTAP databases are extreme lack micro-benchmarks to accurately measure data freshness. Despite the abundance of HTAP databases and benchmarks, there needs to be more thorough research on the performance isolation and real-time analytics capabilities of HTAP databases. This paper focuses on the critical designs of mainstream HTAP databases and the state-of-the-art and state-of-the-practice HTAP benchmarks. First, we systematically introduce the advanced technologies adopted by HTAP databases for real-time analytics and performance isolation capabilities. Then, we summarize the pros and cons of the state-of-the-art and state-of-the-practice HTAP benchmarks. Next, we design and implement a micro-benchmark for HTAP databases, which can precisely control the rate of fresh data generation and the granularity of fresh data access. Finally, we devise experiments to evaluate the performance isolation and real-time analytics capabilities of the state-of-the-art HTAP database. In our continued pursuit of transparency and community collaboration, we will soon make available our comprehensive specifications, meticulously crafted source code, and significant results for public access at https://www.benchcouncil.org/mOLxPBench.

1. Introduction

Hybrid **T**ransactional/**A**nalytical **P**rocessing (HTAP) databases are expected to meet the needs of real-time analytics applications [1–4] because they eliminate the extract-transform-load (ETL) processing between the OLTP database and data warehouse. HTAP databases aim to perform real-time analytics on the fresh data generated by online transactions and mitigate the performance interference between online transactions and analytical queries. To achieve the objectives mentioned above, the mainstream HTAP databases use dual data stores to guarantee performance isolation and optimize the data update propagation between the dual data stores to speed up real-time analytics.

To achieve performance isolation between online transactions and analytical queries, HTAP databases process online transactions in the row-based data store and analytical queries in the column-based data store. HTAP databases optimize the row-based and the column-based data stores, respectively, to speed the execution of online transactions and analytical queries. The row-based data store utilizes indexing and concurrency control mechanisms to facilitate update-intensive online transactions [2,5–8]. In addition, the column-based data store achieves a high compression rate and enhanced access for read-intensive analytical queries [9,10]. HTAP databases generally deploy the row-based and column-based data store on the different data nodes [11–13] to avoid high resource contention between online transactions and analytical queries. This would result in considerable latency when propagating data updates from the row-based to the column-based data store. Consequently, optimizing the data update propagation mechanism is another issue for HTAP databases to address.

Fast data update propagation from the row-based to the column-based data stores is essential for real-time analytics. The latency of data update propagation determines the freshness of the analytical data. The process of data update propagation is divided into three steps. The first step is moving the data update from the row-based to the column-based data stores. The second step is translating the row-format data into column-format data. The last step is merging the delta updates into the column-based data store. HTAP databases optimize one or all of the above steps to improve the freshness of analytical data. For example, TiDB [11] preserves only the committed change log and removes redundant information before translating it

* Corresponding author.
E-mail addresses: kangguoxin@ict.ac.cn (G. Kang), chensimin22z@ict.ac.cn (S. Chen), lihongxiao19@mails.ucas.ac.cn (H. Li).

https://doi.org/10.1016/j.tbench.2023.100122
Received 19 April 2023; Received in revised form 4 July 2023; Accepted 5 July 2023
Available online 8 July 2023

Table 1
The key designs of HTAP databases: can HTAP benchmarks evaluate them?

Benchmark name	Performance isolation		Real-time analytics		Component performance	
	OLTP workloads	OLAP workloads	Fresh data generation rate	Fresh data access granularity	Index mechanism	Query range control
CH-benCHmark	√	√				
HTAPBench	√	√				
CBTR	√	√				
OLxPBench	√	√				
HATtrick	√	√				
ADAPT	√	√			√	
HAP	√	√			√	
Micro-benchmark	√	√	√	√	√	√

into column-format data to decrease data movement. In contrast to TiDB, which deploys the row-based and column-based data stores on separate data nodes, some HTAP databases [9,14–17] deploy the row-based and column-based data stores on the same server to prevent data update propagation across the different data nodes. It slows down the latency of delta updates moving but poses a significant challenge to performance isolation.

Equally as important as it is to track advanced technologies for HTAP databases is to evaluate these HTAP databases. HTAP benchmarks must measure how well the HTAP databases can do performance isolation and real-time analytics. We will introduce the existing HTAP benchmarks from schema design, workload composition, and metrics as shown in Table 1.

Firstly, there are stitched schema and semantically consistent schema. The stitched schema is combined with the TPC-C schema [18] and TPC-H [19] schema. It extracts the New-Order, Stock, Customer, Orderline, Orders, Item, Warehouse, District, and History relationships from TPC-C schema [18] to integrate them with the Supplier, Country, and Region relationships of TPC-H schema [19]. CH-benCHmark [20] proposes the stitched schema, which is followed by HTAPBench [21] and Swarm64 [22]. Analytical queries cannot access the valuable data generated by online transactions and stored in the History table when using the stitched schema. And the stitched schema will affect the semantics of HTAP benchmarks. Therefore, OLxPBench [23] advocates that HTAP benchmarks should employ the semantically consistent schema instead of the stitched schema. The semantically consistent schema emphasizes that online transactions and analytical queries access the same schema. Analytical queries can access all business data generated by online transactions. The semantically consistent schema can thus reveal the performance inference between OLTP and OLAP workloads. CBTR [24,25], OLxPBench [23], HATtrick [26], ADAPT [27], and HAP [28] benchmark all employ semantically consistent schema described in Sections 5 and 6.

Secondly, HTAP benchmarks include OLTP workloads, OLAP workloads, and hybrid workloads. OLTP workloads combine read and write operations, whereas OLAP workloads are read-intensive. Hybrid workload refers to the analytical query performed between online transactions. Existing HTAP benchmarks include OLTP and OLAP workloads to investigate performance inference between them. OLxPBench is the only benchmark that evaluates the true HTAP capability of HTAP databases using hybrid workloads. Complex online transactions and analytical queries have a lot of operations, so it is hard to judge how well each operation works on its own. ADAPT [27] and HAP [28] are Micro-benchmarks for a specific operation. However, the ADAPT and HAP benchmarks only include a handful of typical HTAP workloads. ADAPT and HAP, for instance, include an insufficient number of scan queries to evaluate index performance. Micro-benchmarks should provide point scans, small-range and large-range queries for HTAP database evaluation. There are a few Micro-benchmarks available for HTAP databases.

Thirdly, the metrics of HTAP databases are separated into two categories: throughput metrics and latency metrics. The HTAP database evaluates the throughput of OLTP workloads using the transactions per second (tps) and transactions per minute (tpmC) metrics. The HTAP database evaluates the throughput of OLAP workloads using the queries completed per second (qps) and queries completed per hour (QphH) metrics. CH-benCHmark [20] proposes the metrics $\frac{tpmC}{QphH}@tpmC$ and $\frac{tpmC}{QphH}@QphH$ for evaluating the performance isolation between OLTP and OLAP workloads. The former metric considers online transactions the primary workload, while the latter considers analytical queries the primary workload. In contrast, Anja Bog et al. [26]. establish the HATtrick benchmark, which equalizes transactional and analytical workloads. HATtrick [26] defines the throughput frontier and freshness metrics for measuring performance isolation and data freshness, as specified in Section 5.5. HTAP benchmarks utilize average latency and tail latency metrics in addition to throughput metrics. Average latency is the average time it takes for a transaction/query to be processed, whereas tail latency refers to the high percentile latency. Tail latency is an important metric to consider in HTAP databases where a small number of lengthy transactions/queries can substantially impact overall performance or user experience.

This paper makes the following contributions. (1) We systematically introduce the advanced technologies adopted by HTAP databases for these key designs; (2) We summarize the pros and cons of the state-of-the-art and state-of-the-practice HTAP benchmarks for key designs of HTAP databases; (3) We quantitatively compared the differences between micro-benchmarks and macro-benchmarks in evaluating the real-time analytical capabilities of HTAP databases. Micro-benchmark can control the generation and access granularity of fresh data, enabling precise measurement of real-time analytical capabilities of HTAP databases. (4) We measure the performance of individual components of the HTAP database, such as the indexing mechanism. By isolating specific operations, developers can test the performance of these components under different workloads and configurations, which is the foundation of component optimization.

2. Motivation — Micro-benchmarks can control the rate at which fresh data is generated and the granularity of access, which distinguishes them from macro-benchmarks

HTAP databases are extreme lack the micro-benchmark because there is no open-source micro-benchmark. We design and implement a micro-benchmark to investigate the distinction between the micro-benchmark and the macro-benchmark. We select the state-of-the-art HTAP benchmark OLxPBench as the micro-benchmark comparison object. Micro-benchmark is better suited for real-time analytics evaluation because it precisely controls the rate at which fresh data is generated and the granularity of fresh data access. Micro-benchmark queries typically consist of a single statement. For instance, the analytical query calculates the number of rows within a specified range. This indicates that the computational intensity of analytical queries can be managed by adjusting their computational range. And the transactional query updates the value of the specified column in a random row.

Micro-benchmark can adjust the rate at which fresh data is generated to assess the performance of data update propagation between the transactional and analytical instances. The performance interference between transactional and analytical queries can be disregarded when

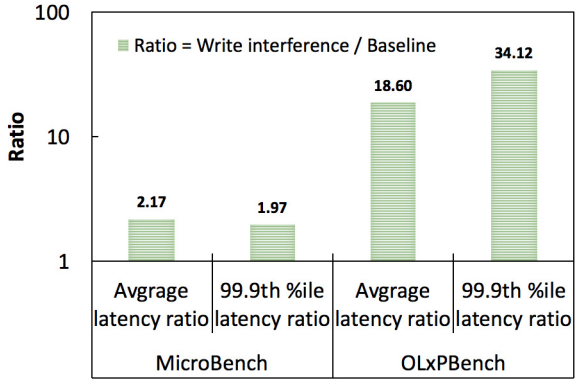

Fig. 1. This figure reveals that the micro-benchmark can accurately measure the real-time analytical capabilities of the HTAP database by controlling read and write interference.

the number of concurrent requests is low. Consequently, almost all of the growing proportion of analytical latency is due to the propagation of data updates. The online transactions and analytical queries in OLxPBench are too complex to control the write and read ranges precisely. The New-Order transaction, for instance, involves numerous inserting and updating operations. The analytical query (Q6) includes operations involving aggregation and sub-selection. This causes the New-Order transaction to generate fresh data that is only partially required by the analytical query (Q6). However, the analytical query must wait for all data updates to propagate before accessing the fresh data. Unlike OLxPBench, micro-benchmark makes it simple to control the rate at which fresh data is generated and the granularity of access to analytical queries on fresh data.

Fig. 1 compares the impact of simple write operations and the New-Order transaction on the measurement of data freshness. The New-Order transaction includes an excessive number of updating and inserting operations, thereby introducing data synchronization that is unnecessary for measuring data freshness. The greater the ratio, the more data needs to be synchronized. It demonstrates that the tail latency of analytical queries (Baseline) increases approximately one-fold when the micro-benchmark is used to simulate write interference. The New-Order transaction contains numerous inserting and updating operations, so the tail latency of the baseline (Q6) increases by more than 36 times when OLxPBench is used. The greater the number of inserting and updating operations, the greater the number of data updates that must be synchronized between transactional and analytical instances. However, not all data updates resulting from online transactions are required for analytical queries. The data freshness measurement will be affected by data updates that are not required by the analytical query. Measuring data freshness requires precise control over the rate of fresh data generation and access granularity.

3. Key designs of mainstream HTAP databases

The mainstream HTAP databases are designed for two objects: real-time analytics and performance isolation. Performance isolation emphasizes that online transactions and analytical queries execute concurrently without affecting each other's performance. Real-time analytics means analyzing the fresh data generated by online transactions as soon as possible. Online transactions and analytical queries can achieve superior isolation performance through the use of independent storage engines. However, the necessity of data synchronization between row-based and column-based storage engines undeniably introduces data synchronization latency. Consequently, real-time access to fresh data during analytical queries becomes a formidable challenge. Therefore, it is challenging for HTAP databases to provide real-time analytics and performance isolation capabilities. Some HTAP databases deploy the transactional and analytical instances on the same server to avoid long turnaround times for delta updates. And other HTAP databases handle online transactions and analytical queries on separate servers to prevent performance interference. This section studies how HTAP databases accomplish real-time analytics and performance isolation.

3.1. Performance isolation

Single-node HTAP databases implement row-based data storage for online transactions and column-based data storage for analytical queries. Because of the intense resource contention, single-node HTAP databases cannot provide performance isolation [29]. Previous works [30,31] have proposed various approaches to partitioned hardware resources to ensure performance isolation. Raza et al. [30] divide the CPU and memory resources into two groups: the first group binds with the specified transactional instance and analytical instance. In contrast, the second group comprises reserved resources assigned based on actual requirements. By dividing the last-level cache (LLC) between the analytical queries and the online transactions, Sirin et al. [31] reduce the performance impact of the analytical queries on the online transactions. Polynesia [15] identifies the root cause of performance interference as the sharing of hardware resources and consequently provides an isolated computing resource for online transactions and analytical queries.

Distributed HTAP databases [11–13] deploy row-based and column-based data stores on separate servers, thereby wholly resolving the issue of resource contention. TiDB [11] implements the $TiKV$ and $TiFlash$ instances for row-based and column-based data stores, respectively. It employs the raft algorithm to replicate asynchronously $TiKV$ logs to $TiFlash$ instances. Due to the collaborative capabilities of Google's internal systems, F1 Lightning [12] contributes a loosely coupled HTAP solution that enables the $F1$ $Query\ engine$ to function with existing OLTP systems and data sources [32–37]. As a result, F1 Lightning [12] need to utilize the $Lightning$ component to capture the data updates from various data sources and translate them into the unified format data.

SingleStore [13] and OceanBase [38] are well-known distributed HTAP databases. They all utilize unified storage to facilitate online transactions and analytical queries. OceanBase [38] demonstrates commendable proficiency in resource isolation. PolarDB-IMCI [39] also provides effective resource isolation for transactional and analytical queries.

3.2. Real-time analytics

Initially, HTAP databases deploy the transactional and analytical instances on a single server to obtain the fresh data generated by online transactions. SAP HANA [9,40] maintains multiple delta update stores for the same table, allowing online transactions updating and existing data updates merging process to be performed in different delta stores. It is permitted for analytical queries to simultaneously access the freshest data in multiple deltas and column-based data stores. SAP HANA [9,40], DB2 BLU [17] and Oracle [14] have implemented an in-memory column-based data store for fast analytics. DB2 [17] supports HTAP workloads with BLU acceleration. Oracle [14] and DB2 BLU [17] make use of numerous analytical optimization technologies, including compression and single-instruction multiple-data (SIMD). It cannot update column data in real-time because data updates are only merged to the column-based data store when the ratio of data updates exceeds a certain threshold.

With the growing amount of real-time data, the single-node HTAP database cannot meet the high scalability and availability requirements. Oracle, for instance, releases a new distributed version that provides scale-out compute and storage resources and implements a

real-time column-based data duplication mechanism for high availability requirements [41]. And then comes the issue of data update propagation. Because data updates must propagate from transactional servers to analytical servers, it is challenging for distributed HTAP databases to guarantee that analytical queries can access the most recent data updates [42]. Both TiDB and F1 Lightning have specialized components for data update propagation. TiDB [11] utilizes the $Logreplication$ process asynchronously to maintain data consistency between $TiKV$ and $TiFlash$ instances. The $Changepump$ component of F1 Lighting [12] provides a consistency protocol that enables analytical queries to access in-memory delta updates generated by online transactions immediately. Once a system failure occurs, the in-memory delta updates are recoverable through the transactional log.

4. Can existing HTAP benchmarks evaluate the key design of HTAP databases?

HTAP databases provide performance isolation for OLTP workloads and OLAP workloads while ensuring that OLAP workloads have access to fresh data generated by OLTP workloads. Evaluating the performance of individual HTAP database components is paramount in optimizing their efficiency.

Table 1 summarizes the existing HTAP benchmarks. Currently, open-source HTAP benchmarks are macro-benchmarks, encompassing intricate online transactions and analytical queries. This approach facilitates a comprehensive assessment of the performance isolation capabilities inherent in HTAP databases. However, due to the extensive number of statements within online transactions and analytical queries, accurately evaluating the performance of specific HTAP database components, such as index performance, presents a considerable challenge. Benchmarking can utilize range scan queries to measure the performance of various index mechanisms in HTAP databases. For instance, theoretically, point queries can effectively evaluate the performance of Hash indexes and LSM-Tree indexes. In an ideal scenario, a Hash index only needs to perform a single hash computation on the primary key to find the corresponding record, while an LSM-Tree index, using a binary search algorithm, requires multi-level searching. However, since Hash indexes are unordered, handling range queries necessitates scanning the entire index space, in contrast, the ordered LSM-Tree indexes perform more efficiently during range queries. Hence, range scan queries can effectively test and compare the performance of different index mechanisms. In this research, the impact of query scope on computational workload intensity is meticulously investigated. By strategically manipulating the scope of a query, it is possible to exert greater control over the computational demands of the workload. A comparative analysis is performed, examining point queries, small-range queries, and large-range queries, all with identical transmission rates. The results reveal distinct discrepancies in the required computational resources and the subsequent performance outcomes for each query type. This study offers valuable insights into optimizing query execution and enhancing system performance.

Concurrently, the complexity of workloads makes it arduous to regulate the generation rate and access the granularity of fresh data. This predicament leads to measurement biases concerning data freshness. Section 2 elucidates the distinctions between macro-benchmarks and micro-benchmarks in the context of controlling fresh data. Micro-benchmarks are indispensable for appraising the real-time analysis capabilities of HTAP databases. Despite their importance, there is a notable absence of open-source micro-benchmarks explicitly designed for HTAP databases. Consequently, there is an urgent need within the industry and academia to develop tailor-made micro-benchmarks specifically intended for HTAP databases.

5. Macro-benchmarks for HTAP databases

5.1. CH-benchmark

5.1.1. Schema design

CH-benCHmark [20] combines the TPC-C [18] and TPC-H [19] schema. The CH-benCHmark schema retains all TPC-C tables and adds the Supplier, Nation, and Region tables of TPC-H. Fig. 2 depicts the relationships between nine tables.

5.1.2. Workload description

CH-benCHmark provides both online transactions and analytical queries. The OLTP workloads are the same as the TPC-C transactions which are New-Order, Payment, Order-Status, Delivery, and Stock-Level transactions. The default percentages for the aforementioned five transactions are 44%, 44%, 4%, 4%, and 4%, respectively. Order-Status and Stock-Level are read-only transactions, and the remaining three are update-intensive transactions. The 22 analytical queries in CH-benCHmark are derived from the TPC-H benchmark. Analytical queries retain the majority of business semantics but make adjustments based on the CH-benCHmark schema.

5.1.3. Evaluation and metrics

CH-benCHmark evaluates the OLTP, OLAP, and mixed performance of HTAP databases. It regulates the rate at which online transactions and analytical queries are sent by setting the number of request-sending threads. It measures the performance of HTAP databases using response time and throughput metrics. The transactions per minute (tpmC) metric is utilized to measure the throughput of OLTP workloads. And the queries per hour (QphH) metric is utilized to measure the throughput of OLAP workloads. CH-benCHmark inventively designs the $\frac{tpmC}{QphH}@tpmC$ and $\frac{tpmC}{QphH}@QphH$ metrics to measure the performance of mixed workloads that consist of both OLTP and OLAP workloads. $@tpmC$ indicates OLTP as the dominant workload, whereas $@QphH$ indicates OLAP as the dominant workload.

For instance, as shown in expression (1), when transactional and analytical workloads are executed sequentially, their respective throughputs are assumed to be 5084 tpmC and 895.6 QphH.

$$P_1(OLTP) = \frac{5084tpmC}{895.6QphH}@5084tpmC. \tag{1}$$

As shown in expression (2), when executing mixed workloads concurrently, the OLTP and OLAP throughputs are assumed to be 5188 tpmC and 804.2 QphH, respectively.

$$P_2(OLTP) = \frac{5188tpmC}{804.2QphH}@5188tpmC. \tag{2}$$

$P_1(OLTP)$ equals $5.7@5084tpmC$, which is less than $6.5@$ $5188tpmC$ of $P_2(OLTP)$. The results indicate that analytical queries do not hinder the performance of online transactions in this experiment. In addition, CH-benCHmark is the first HTAP benchmark that defines data freshness. CH-benchmark decides whether to use the most recent data for analytical queries by setting either a time threshold or a number of transactions.

5.2. HTAPBench

HTAPBench [21] adopts the same schema as CH-benCHmark, which integrates TPC-C and TPC-H schema. HTAPBench takes five online transactions from TPC-C and 22 analytical queries from TPC-H.

The most distinct aspect between HTAPBench and CH-benCHmark is that HTAPBench proposes a unified metric for HTAP databases, as shown in the expression (3).

$$QpHpW = \frac{QphH}{\#OLAPworkers}@tpmC. \tag{3}$$

QpHpW represents the analytical queries completed in an hour per analytical worker. When the throughput of online transactions remains

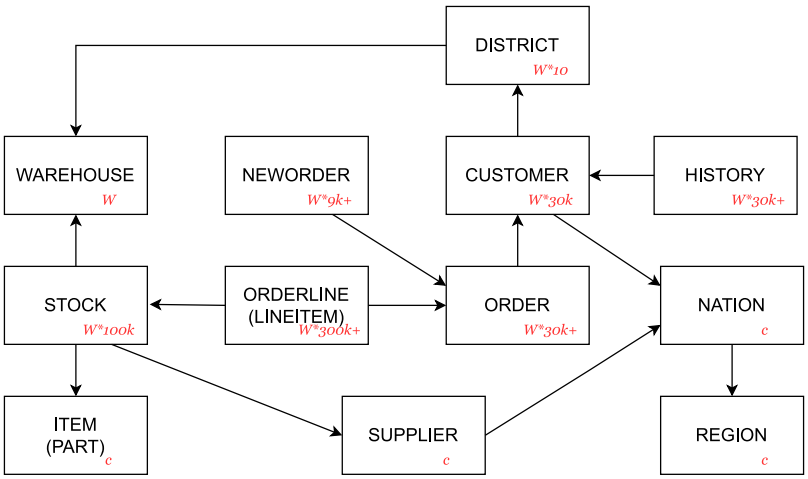

Fig. 2. The schema of CH-benCHmark.

constant, the greater the number of analytical queries completed per hour per analytical worker, the better the performance of the HTAP database.

5.3. CBTR

CBTR [24,25] is the first HTAP benchmark that adopts the semantically consistent schema. The semantically consistent schema, unlike the stitched schema, allows OLTP and OLAP workloads to operate on the same tables as opposed to separate tables for each workload. The schema of CBTR includes 18 tables, which are extracted from the real-world order-to-cash scenario. The normalization of CBTR's schema is configurable, with 1NF being the default. Different normalization levels produce varying degrees of data redundancy, which has a direct impact on the total number of columns. For instance, the schema with the highest degree of redundancy contains 2316 columns in total.

CBTR provides four online read-update transactions, three online read-only transactions, and four online analytical queries. CBTR utilizes data from actual business scenarios rather than synthetic data generated by data generators. However, CBTR is not widely recognized due to its closed-source nature.

5.4. OLxPBench

5.4.1. Schema design

The OLxPBench suite [23] proposes creatively that HTAP benchmarks necessitate a semantically consistent schema. Semantically consistent schema emphasizes that online transactions and analytical queries should use the same data. There are three benchmarks in the OLxPBench suite: subenchmark for general scenarios, fibenchmark for financial scenarios, and tabenchmark for telecom scenarios. Subenchmark reuses the schema of the TPC-C [18], which consists of nine tables. The schema of fibenchmark is derived from that of SmallBank benchmark [43] and has three tables: $ACCOUNT$, $SAVING$, and $CHECKING$ tables. TATP [44], which has four tables, including $SUBSCRIBER$, $SPECIAL\ FACILITY$, $ACCESS\ INFO$, and $CALL\ FORWARDING$ tables, is the source of inspiration for tabenchmark. Tabenchmark modifies the SUBSCRIBER table by expanding a composite primary key.

5.4.2. Workload description

The OLxPBench benchmark suite consists of 18 online transactions, 18 analytical queries, and 17 hybrid transactions. The online transactions of the original OLTP benchmarks remain the same. Just 8% of online transactions in Subenchmark are read-only. 15% of online transactions in Fibenchmark are read-only. 80% of online transactions in Tabenchmark are read-only. In addition, it increases the analytical queries and hybrid transactions based on the semantically consistent schema. The analytical queries consist of complicated analytical statements like aggregation and multi-join. The hybrid transaction incorporates an analytical statement into an online transaction. Read-only hybrid transactions make up 60%, 20%, and 40% of the subenchmark, fibenchmark, and tabenchmark, respectively.

5.4.3. Evaluation and metrics

OLxPBench suite evaluates the performance isolation between the online transactions and analytical queries. It begins by determining the peak throughput of online transactions and analytical queries. Fix the request send rate for online transactions or analytical queries x_f, and progressively increase the request send rate for the other instances x_i. If the throughput and latency of x_f vary minimally, there is no performance interference between online transactions and analytical queries; contrarily, the higher the fluctuation in performance of x_f, the greater the performance interference between online transactions and analytical queries. In addition, the OLxPBench suite evaluates the HTAP performance of HTAP databases using hybrid transactions. Moreover, the scalability of HTAP databases is evaluated.

5.5. HATtrick

5.5.1. Schema design

The schema of the HATtrick benchmark [26] is modified based on Star-Schema Benchmark (SSB) [45]. The schema of the HATtrick benchmark newly adds the HISTORY and FRESHNESS table and appends new attributes to the CUSTOMER, SUPPLIER, and PART table The schema consists of seven tables, as shown in Fig. 3.

5.5.2. Workload description

HATtrick includes both online transactions and analytical queries. It offers three online transactions comparable to the TPC-C benchmark. The transactional workloads consist of 48 percent New-Order, 48 percent Payment, and 4 percent Count orders. The New-order and Payment transactions are update-intensive, while the Count orders transaction is read-only.

Thirteen analytical queries are derived from the SSB benchmark and modified slightly to conform with the schema.

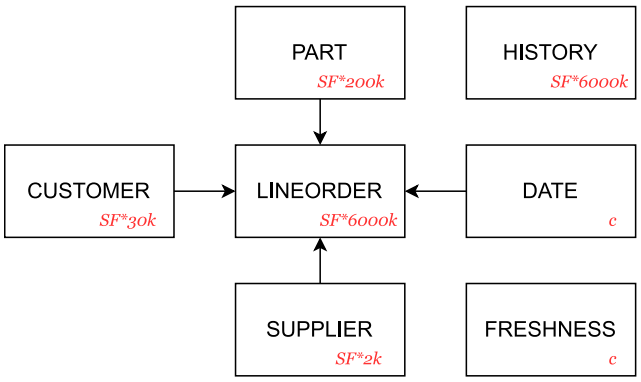

Fig. 3. The schema of SSB.

5.5.3. Evaluation and metrics

HATtrick proposes the throughput frontier and freshness score metrics to measure the performance isolation and data freshness of HTAP databases. The throughput frontier is visualized as a curve with the transactional throughput x_t on the horizontal axis and the analytical throughput y_a on the vertical axis. The maximum transactional throughput is X_t, and the maximum analytical throughput is Y_a. The line formed by the coordinates (0, Y_a), (X_t, Y_a), and (X_t, 0) is the bounding line. And the line formed by the coordinates (0, Y_a) and (X_t, 0) is the proportional line. If the throughput frontier is close to the bounding line, HTAP database performance isolation is stable. If the throughput frontier is below the proportional line, it indicates a significant performance interference between online transactions and analytical queries. The freshness score metric refers to the delay when analytical queries can access the latest data generated by online transactions.

5.6. Advantages and disadvantages

Evaluations of HTAP databases require the proper schema, workloads, and metrics. Ch-benCHmark and HTAPBench are the first HTAP benchmarks to implement the stitched schema, separated online transactions and analytical queries, and metrics described in Section 5.1. CBTR, OLxPBench suite, and HATtrick implement a semantically consistent schema to analyze the performance isolation between online transactions and analytical queries. The CBTR schema is derived from the actual production environment. The OLxPBench suite implements domain-specific benchmarks and innovative hybrid transactions. HATtrick contributes the throughput frontier and freshness score metrics.

6. Micro-benchmarks for HTAP databases

6.1. ADAPT

ADAPT [27] is a synthetic benchmark that extracts typical operations from the production environment [46]. The schema contains both narrow and wide tables. The narrow table contains 50 columns, and the wide table contains 500 columns. ADAPT benchmark contributes five queries: insert query, scan query, maximum aggregate query, sum aggregate query, and join query. ADAPT benchmark lacks delete, update, and point scan queries.

6.2. HAP

Based on the ADAPT benchmark, the HAP benchmark [28] reduces the number of columns in narrow and wide tables. The narrow table has 16 columns, whereas the wide table has 160 columns. HAT benchmark contains six queries: point query, count aggregate query, sum aggregate query, insert query, delete query, and update query. The delete, update, and point scan queries have recently been added to the HAP benchmark. At the same time, it deletes the scan and the join queries.

6.3. Advantages and disadvantages

The ADAPT [27] and HAP [28] benchmarks abstract the basic HTAP operations. However, they contain a limited number of typical HTAP workloads and are not open-source. The micro-benchmark should provide a variety of scan queries, including point queries, small-range queries, and large-range queries. The variety of scan queries is crucial for the index optimization of HTAP databases. In addition, micro-benchmarks must ensure that the read and write operations access the same columns to evaluate the date update propagation capability.

7. Micro-benchmark

7.1. Range setting method

Informed by a theoretical framework, the parameters of a scan query range are thoughtfully established. We start by supposing that the total count of records in a table is represented as S. The range for this operation is defined between two integer values, a lower bound L, and an upper bound U. As described in Eqs. (4) and (5), the boundaries of L and U are established with the essential stipulation that L must always remain less than U. The desired range for our scan query is the calculated difference between U and L. Our ultimate aim is to determine the average range, and then to utilize this average as a pivotal point. Upon establishing this pivotal point, we then select scatter points of several orders of magnitude beneath it to determine the scan query range.

A key strategy in achieving this objective involves a recalibration of the range values, effecting a transformation into a summation of multiple terms by increasing them by 1. This process is illustrated in Eq. (6). Following this, we examine the pattern of the data, accumulate the range values, and then divide this sum by the number of range values. This leads us to the determination of the average range, as represented in Eq. (7). Our calculations reveal that the proximate value of the average range in this random configuration is equivalent to one-third of S, as demonstrated in Eq. (8). Based on these findings, we establish the range of the scan query to fall within the parameters of 0.5% and 10% of the total record count, S. Aggregate and scan queries exhibit the capacity to meticulously regulate the granularity of access to fresh data by expertly delineating the scope of the inquiry. This stands as a distinguishing hallmark, setting micro-benchmarks apart from conventional HTAP benchmarks.

$$L \in [1, S-2] \cap \mathbb{Z}. \tag{4}$$

$$U \in [L+1, S] \cap \mathbb{Z}. \tag{5}$$

$$avgScanSize = \frac{-1 + \sum_{x=1}^{S-1} x(S-x)}{-1 + \sum_{x=1}^{S-1} x}, \tag{6}$$

$$= \frac{\frac{1}{6}(S-1)S(S+1) - 1}{\frac{1}{2}(S-1)S - 1}, \tag{7}$$

$$\approx \frac{1}{3}S. \tag{8}$$

7.2. The design and implementation

There is no open-source micro-benchmark for HTAP databases. The micro-benchmark could precisely regulate the read/write ratio for a comprehensive evaluation of HTAP databases. Therefore, we mimic the ADAPT and HAP benchmarks to design and implement the micro-benchmark, which accomplishes the six queries listed below. Moreover, the micro-benchmark contains a single table with 59 attributes named $ITEM$. The attributes in the $ITEM$ table are derived from the actual e-commerce applications.

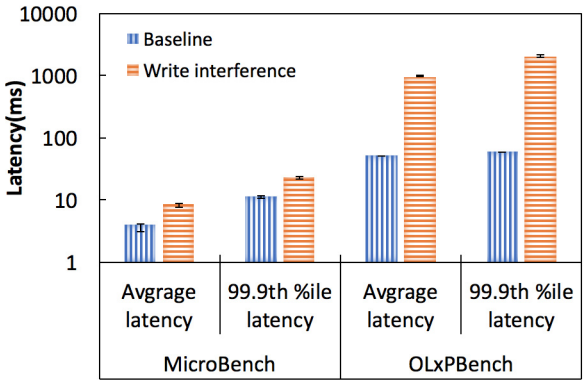

Fig. 4. This figure illustrates the distinction between the micro-benchmark and OLxPBench, the state-of-the-art HTAP benchmark.

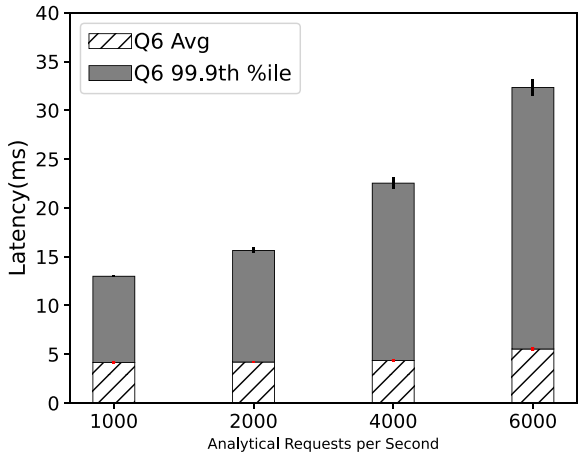

Fig. 5. Small aggregate query performance.

Q1 is a point-get query retrieves the record where the primary key equals a random number. Q4 is a small-range scan query that randomly retrieves 0.5% of the records. Q5 is a large-range scan query that randomly retrieves 10% of the records. Q3 is an update query that updates the specific value of a random record. HTAP database indexing and writing speeds can be measured with Q1, Q4, Q5, and Q3. Q2 is an aggregate query that counts the records in a random range. Q6 is a small-range aggregate query that counts 0.5% of the records. Q2 and Q6 are useful to measure the OLAP performance of HTAP databases. All experimental results reported in this paper are the mean and standard deviation of five independent runs.

Q1: **SELECT** $i_1, i_2, \ldots, i_k$ **FROM** ITEM
WHERE $i_{id} = v$;

Q2: **SELECT COUNT**$(*)$ **FROM** ITEM
WHERE $i_{id} \in [v_s, v_e]$;

Q3: **UPDATE** ITEM **SET** $i_r = v_r$
WHERE $i_{id} = v_s$;

Q4: **SELECT** $i_1, i_2, \ldots, i_k$ **FROM** ITEM
WHERE $i_{id} \in [v_s, v_p]$;

Q5: **SELECT** $i_1, i_2, \ldots, i_k$ **FROM** ITEM
WHERE $i_{id} \in [v_s, v_q]$;

Q6: **SELECT COUNT**$(*)$ **FROM** ITEM
WHERE $i_{id} \in [v_s, v_q]$;

8. Evaluation

8.1. Experimental setup

8.1.1. Environment

The server node has two Intel Xeon E5-2699v4@2.20 GHz CPUs, 128 GB memory, and two 2TB SSD. The client node has two Intel Xeon E5645@2.40 GHz CPUs, 48 GB memory, and eight 2TB HDDs. The server and client run on Ubuntu 20.04 version and are connected by a 10 Gbps Ethernet network.

8.1.2. Database

TiDB is an industry-standard HTAP database, and its version is 6.1.0. We deploy the *tidb* instance, the *TiKV* instance, and the *TiFlash* instance on the same server in order to evaluate the real-time analytics and performance isolation capabilities in depth. The components of TiDB are described in detail in Section 3.

8.2. Comparing micro-benchmark to the state-of-the-art HTAP benchmark

As the experimental workload, we selected the New-Order transaction and analytical query (Q6) from the Subenchmark in the OLxPBench suite. In addition, we utilize Q2 and Q3 as experimental workloads. Each experiment is performed five times independently, and the mean and standard deviation are reported. To avoid performance interference between transactional and analytical instances, concurrent requests are limited to 100 transactional and analytical requests per second. As depicted in Fig. 4, the average and tail latency of Q2 nearly doubles with interference from Q3. Due to the interference of the New-Order transaction, the average and tail latency of Q6 increased by 19 and 34 times, respectively. This is because the New-Order transaction in OLxPBench contains an excessive number of inserting and updating operations, resulting in an excessive number of data updates to propagate. However, not all New-Order data update records are required by the analytical query (Q6). In micro-benchmark, Q2 requires all data updates produced by Q3. Unnecessary data updates introduce excessive synchronization latency, resulting in inaccurate data freshness measurements. The premise of measuring data freshness is therefore to strictly control the generation rate and gain access to the granularity of fresh data.

8.3. Scan performance

We set up the point query, small range query, and large range query to fully evaluate the index performance of HTAP databases. The scan performance is shown in Fig. 9, Fig. 11, and Fig. 10. The diagonal area represents the average latency, while the area with the gray shading represents the tail latency. Every experiment is shown in this manner and will not be discussed below. Peak throughput for point query, small-range scan query, and large-range scan query exceed 20,000, 10,000, and 400 tps, respectively. Peak throughput decreases as the total of scan records expands. The average latency of the point query illustrated in Fig. 9 is less than five milliseconds. The average latency of the small-range scan query illustrated in Fig. 11 is less than ten milliseconds. *TiKV* has implemented a scalable, ordered LSM-Tree index, with experimental results demonstrating that the latency for both point queries and small-range scan queries is within the millisecond level. When handling point queries, TiDB performs binary searches based on the primary key's value, necessitating multi-level searching to locate the relevant data block. Furthermore, TiDB has parallel optimizations for range queries, using a Coprocessor to concurrently access ordered blocks, thereby accelerating the processing speed for range queries.

The average latency of the large-range scan query illustrated in Fig. 10 is the greatest and exceeds twenty milliseconds. And the 99.9th

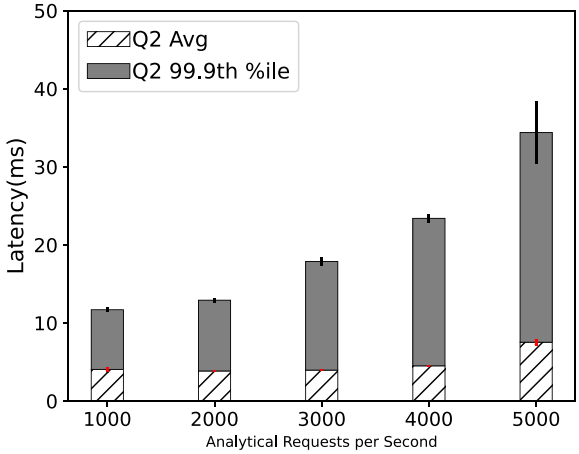

Fig. 6. Random aggregate query performance.

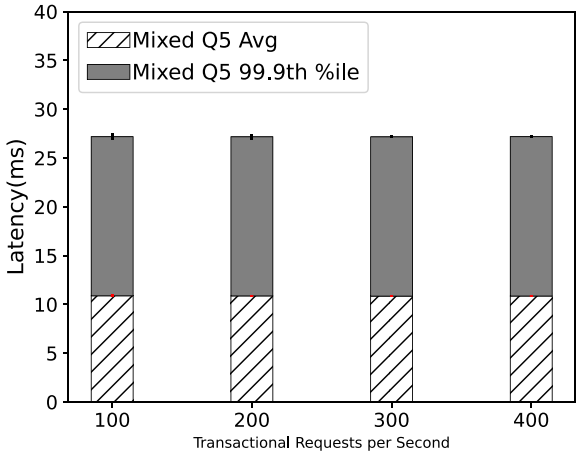

Fig. 7. Performance interference of Q5 on Q2.

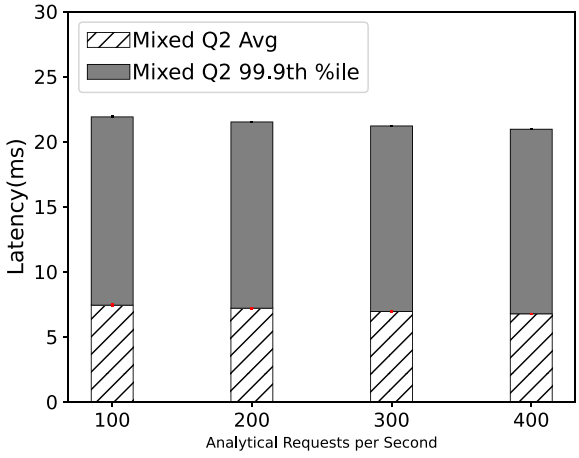

Fig. 8. Performance interference of Q2 on Q5.

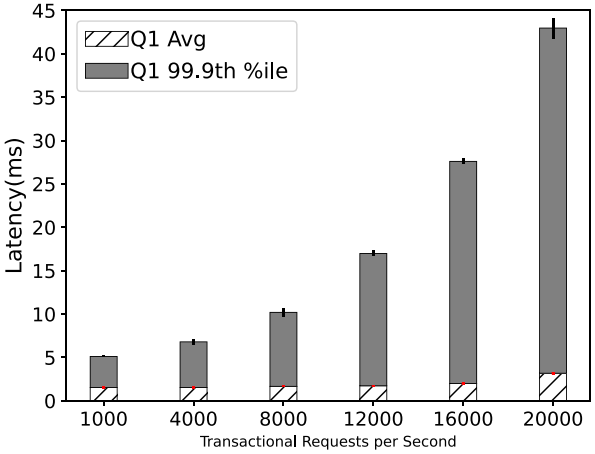

Fig. 9. Point-get query performance.

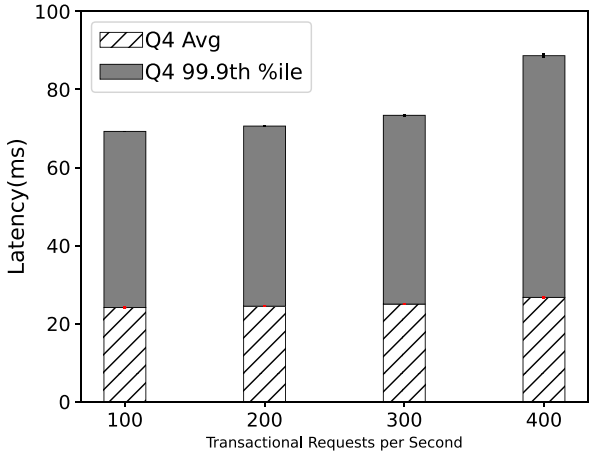

Fig. 10. Large-range scan query performance.

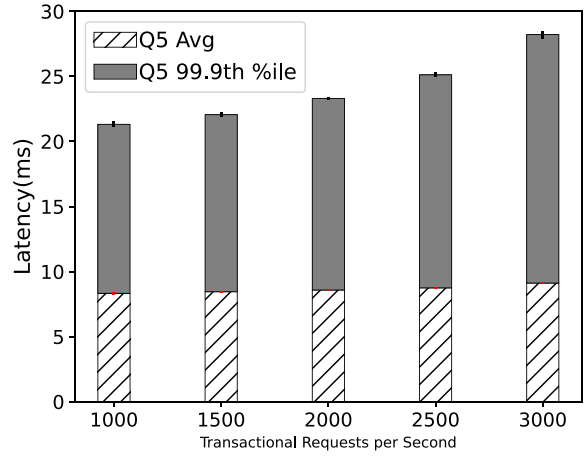

Fig. 11. Small-range scan query performance.

percentile latency of the large-range scan query is greater than 45 ms. The point query retrieves the targeted record by primary key. The range queries push the task down to $TiKV$ instance execution and summarize the $TiKV$ instance return results in the SQL engine. In addition, range scan queries retrieve the continuous records stored in a small number of regions, which can lead to access hotspot issues.

8.4. Update performance

We use the update queries that follow the uniform distribution to measure the update performance. The performance results of the update operation are depicted in Fig. 12. The average latency increases 1.9× with the transactional requests increasing from 1000 tps to 10000 tps. Meanwhile, the 99.9th percentile latency increases from 10.4 ms

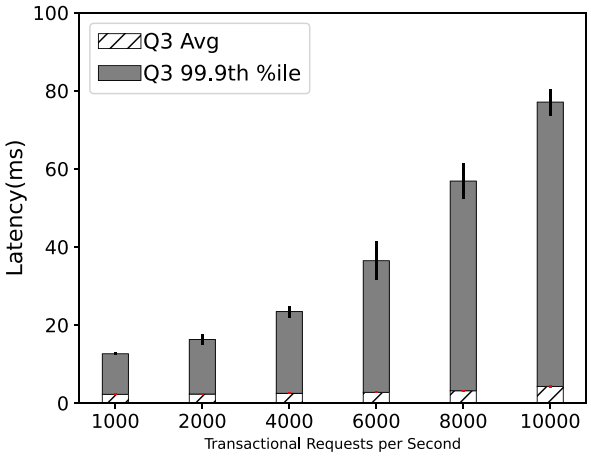

Fig. 12. Update query performance.

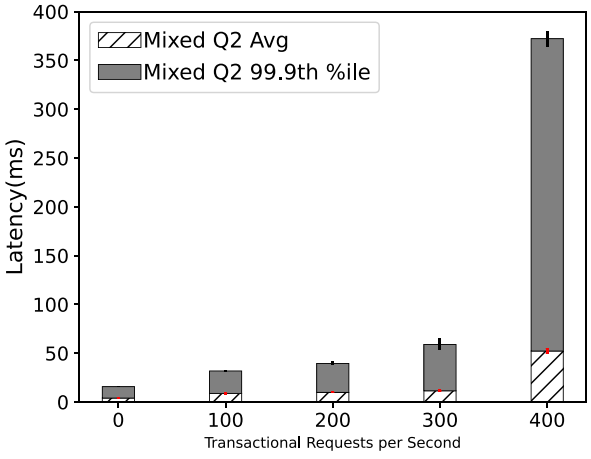

Fig. 13. Real-time analytic performance.

to 72.8 ms. In the experiment, write conflicts gradually appear as the number of concurrent update requests increases. The reason for this experimental phenomenon is that TiDB uses the pessimistic model by default, and the "autocommit" option is enabled. A transaction is initially committed as an optimistic transaction and then, if a write conflict occurs, as a pessimistic transaction. If there are violent write conflicts, it is recommended to disable "autocommit" option.

8.5. Aggregate performance

Q2 and Q6 return the number of rows retrieved. When analytical requests per second are less than 4000 tps, the average latency of Q2 and Q6 is around four milliseconds. As shown in Figs. 5 and 6, the maximum 99.9th percentile latency for Q2 is 34.39 ms, and that for Q6 is 26.79 ms. Q6 has a greater peak throughput and a shorter tail latency than Q2 due to the fact that Q2 requires more calculations. For data aggregation, TiDB implements the hash aggregation operator and the stream aggregation operator. The SQL engine uses the stream aggregation operation to deal with the COUNT(*) function. The stream aggregation operator requires less memory than the stream aggregation operator.

8.6. Hybrid performance

8.6.1. Performance isolation evaluation

To investigate the performance isolation issue, we deploy the $TiKV$ and $TiFlash$ instances on the same server. We employ read-only queries for the performance isolation evaluation to prevent the interference of data update propagation. $TableRangeScan$ and $StreamAgg$ are the operators utilized for the large-range scan and aggregate queries, respectively. The send rate of the Q5 remains constant while the sending rate of the aggregate inquiries steadily increases from 100 tps to 400 tps in the first set of experiments, as shown in Fig. 7. The send rate of the Q2 remains constant in the second set of experiments, as shown in Fig. 8, while the sending rate of the scan queries steadily increases from 100 tps to 400 tps. As the sending rate of interference inquiries rises, the latency of scan and aggregate queries remains relatively constant within the error bounds. Even when placed on the same server, there is not much performance interference between $TiKV$ and $TiFlash$ instances when there is no resource competition between mixed workloads.

8.6.2. Real-time analytic evaluation

We deploy the $TiKV$ and $TiFlash$ instances in the same server to guarantee the analytical query analyzes the fresh transactional data as soon as possible. To minimize interference, the $TiKV$ and $TiFlash$ instances are deployed on the different solid-state drives. We keep the analytical requests per second constant, increasing the proportion of updated data to guarantee an increasing proportion of fresh data that analytical requests can access. As shown in Fig. 13, the number of update requests per second rises from 100 to 400 tps. The send rate of the analytical queries remains constant, and we collect the analytical queries' latency results. A low number of concurrent requests avoids performance interference between update and aggregate queries, which is demonstrated in Section 8.6.1. Both the average and 99.9th percentile latency rise by more than a factor of one due to the propagation of data updates. The average latency of analytical queries is 52.26 ms, and the 99.9 percentile delay is 320.04 ms when the send rate of update queries is 400 tps. The aforementioned performance results indicate that TiDB can complete real-time analytics in 500 ms without transferring data updates across nodes.

9. Conclusion

This paper involves a thorough introduction of HTAP database strategies for enhancing performance isolation and real-time analytics. In addition, we compare state-of-the-art and best-practice HTAP benchmarks in terms of schema model, workloads, and evaluation metrics. The CBTR, OLxPBench, HATtrick, ADAPT, and HAP benchmarks all use the semantically consistent schema. OLxPBench is innovative and provides a hybrid transaction that executes the analytical statement between the online transaction. And HATtrick contributes the throughput frontier and freshness metrics.

Currently, HTAP databases are severely lacking in micro-benchmarks to precisely manage read and write ranges. Consequently, we implement a micro-benchmark in Section 7 to measure the performance isolation and real-time analytics capabilities of HTAP databases. When the number of concurrent requests is modest, the performance of transactional and analytical instances on the same server does not interfere with one another. The propagation of data updates on the same node promotes the preservation of analytical data's freshness. Moreover, rigorous resource partitioning between transactional and analytical instances may facilitate dual-format HTAP databases to support both performance isolation and real-time analytics.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Ronald Barber, Christian Garcia-Arellano, Ronen Grosman, Rene Mueller, Vijayshankar Raman, Richard Sidle, Matt Spilchen, Adam J. Storm, Yuanyuan Tian, Pinar Tözün, et al., Evolving databases for new-gen big data applications, in: CIDR, 2017.

[2] Chen Luo, Pinar Tözün, Yuanyuan Tian, Ronald Barber, Vijayshankar Raman, Richard Sidle, Umzi: Unified multi-zone indexing for large-scale HTAP, in: Advances in Database Technology-22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26–29, 2019, OpenProceedings. org, 2019, pp. 1–12.

[3] Fatma Özcan, Yuanyuan Tian, Pinar Tözün, Hybrid transactional/analytical processing: A survey, in: Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 1771–1775.

[4] Hemant Saxena, Lukasz Golab, Stratos Idreos, Ihab F. Ilyas, Real-time LSM-trees for HTAP workloads, 2021, arXiv preprint arXiv:2101.06801.

[5] Yihan Sun, Guy E. Blelloch, Wan Shen Lim, Andrew Pavlo, On supporting efficient snapshot isolation for hybrid workloads with multi-versioned indexes, Proc. VLDB Endow. 13 (2) (2019).

[6] R. Malinga Perera, Bastian Oetomo, Benjamin I.P. Rubinstein, Renata Borovica-Gajic, No DBA? No regret! multi-armed bandits for index tuning of analytical and HTAP workloads with provable guarantees, 2021, arXiv preprint arXiv:2108.10130.

[7] Jinwei Guo, Peng Cai, Jiahao Wang, Weining Qian, Aoying Zhou, Adaptive optimistic concurrency control for heterogeneous workloads, Proc. VLDB Endow. 12 (5) (2019) 584–596.

[8] Tobias Vinçon, Christian Knödler, Leonardo Solis-Vasquez, Arthur Bernhardt, Sajjad Tamimi, Lukas Weber, Florian Stock, Andreas Koch, Ilia Petrov, Near-data processing in database systems on native computational storage under htap workloads, Proc. VLDB Endow. 15 (10) (2022) 1991–2004.

[9] Franz Färber, Sang Kyun Cha, Jürgen Primsch, Christof Bornhövd, Stefan Sigg, Wolfgang Lehner, SAP HANA database: data management for modern business applications, ACM SIGMOD Rec. 40 (4) (2012) 45–51.

[10] Michael Abebe, Horatiu Lazu, Khuzaima Daudjee, Proteus: Autonomous adaptive storage for mixed workloads, in: Proceedings of the 2022 International Conference on Management of Data, 2022, pp. 700–714.

[11] Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Li Shen, Liu Tang, Yuxing Zhou, Menglong Huang, et al., TiDB: a Raft-based HTAP database, Proc. VLDB Endow. 13 (12) (2020) 3072–3084.

[12] Jiacheng Yang, Ian Rae, Jun Xu, Jeff Shute, Zhan Yuan, Kelvin Lau, Qiang Zeng, Xi Zhao, Jun Ma, Ziyang Chen, et al., F1 lightning: HTAP as a service, Proc. VLDB Endow. 13 (12) (2020) 3313–3325.

[13] Adam Prout, Szu-Po Wang, Joseph Victor, Zhou Sun, Yongzhu Li, Jack Chen, Evan Bergeron, Eric Hanson, Robert Walzer, Rodrigo Gomes, et al., Cloud-native transactions and analytics in SingleStore, in: Proceedings of the 2022 International Conference on Management of Data, 2022, pp. 2340–2352.

[14] Tirthankar Lahiri, Shasank Chavan, Maria Colgan, Dinesh Das, Amit Ganesh, Mike Gleeson, Sanket Hase, Allison Holloway, Jesse Kamp, Teck-Hua Lee, et al., Oracle database in-memory: A dual format in-memory database, in: 2015 IEEE 31st International Conference on Data Engineering, IEEE, 2015, pp. 1253–1258.

[15] Amirali Boroumand, Saugata Ghose, Geraldo F. Oliveira, Onur Mutlu, Polynesia: Enabling high-performance and energy-efficient hybrid transactional/analytical databases with hardware/software co-design, in: 2022 IEEE 38th International Conference on Data Engineering, ICDE, 2022, pp. 2997–3011.

[16] Per-Åke Larson, Adrian Birka, Eric N. Hanson, Weiyun Huang, Michal Nowakiewicz, Vassilis Papadimos, Real-time analytical processing with SQL server, Proc. VLDB Endow. 8 (12) (2015) 1740–1751.

[17] Vijayshankar Raman, Gopi Attaluri, Ronald Barber, Naresh Chainani, David Kalmuk, Vincent KulandaiSamy, Jens Leenstra, Sam Lightstone, Shaorong Liu, Guy M. Lohman, et al., DB2 with BLU acceleration: So much more than just a column store, Proc. VLDB Endow. 6 (11) (2013) 1080–1091.

[18] TPC-C Benchmark, 2010.

[19] TPC-H Benchmark, 2010.

[20] Richard Cole, Florian Funke, Leo Giakoumakis, Wey Guy, Alfons Kemper, Stefan Krompass, Harumi Kuno, Raghunath Nambiar, Thomas Neumann, Meikel Poess, et al., The mixed workload CH-benCHmark, in: Proceedings of the Fourth International Workshop on Testing Database Systems, 2011, pp. 1–6.

[21] Fábio Coelho, João Paulo, Ricardo Vilaça, José Pereira, Rui Oliveira, Htapbench: Hybrid transactional and analytical processing benchmark, in: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, 2017, pp. 293–304.

[22] Swarm64 HTAP Benchmark for PostgreSQL, 2021.

[23] Guoxin Kang, Lei Wang, Wanling Gao, Fei Tang, Jianfeng Zhan, OLxPBench: Real-time, semantically consistent, and domain-specific are essential in benchmarking, designing, and implementing HTAP systems, in: 2022 IEEE 38th International Conference on Data Engineering, ICDE, 2022, pp. 1822–1834.

[24] Anja Bog, Kai Sachs, Hasso Plattner, Interactive performance monitoring of a composite oltp and olap workload, in: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, 2012, pp. 645–648.

[25] Anja Bog, Hasso Plattner, Alexander Zeier, A mixed transaction processing and operational reporting benchmark, Inf. Syst. Front. 13 (2011) 321–335.

[26] Elena Milkai, Yannis Chronis, Kevin P. Gaffney, Zhihan Guo, Jignesh M. Patel, Xiangyao Yu, How good is my HTAP system? in: Proceedings of the 2022 International Conference on Management of Data, 2022, pp. 1810–1824.

[27] Joy Arulraj, Andrew Pavlo, Prashanth Menon, Bridging the archipelago between row-stores and column-stores for hybrid workloads, in: Proceedings of the 2016 International Conference on Management of Data, 2016, pp. 583–598.

[28] Manos Athanassoulis, Kenneth S. Bøgh, Stratos Idreos, Optimal column layout for hybrid workloads, Proc. VLDB Endow. 12 (13) (2019) 2393–2407.

[29] Iraklis Psaroudakis, Florian Wolf, Norman May, Thomas Neumann, Alexander Böhm, Anastasia Ailamaki, Kai-Uwe Sattler, Scaling up mixed workloads: a battle of data freshness, flexibility, and scheduling, in: Performance Characterization and Benchmarking. Traditional to Big Data: 6th TPC Technology Conference, TPCTC 2014, Hangzhou, China, September 1–5, 2014. Revised Selected Papers 6, Springer, 2015, pp. 97–112.

[30] Aunn Raza, Periklis Chrysogelos, Angelos Christos Anadiotis, Anastasia Ailamaki, Adaptive HTAP through elastic resource scheduling, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020, pp. 2043–2054.

[31] Utku Sirin, Sandhya Dwarkadas, Anastasia Ailamaki, Performance characterization of htap workloads, in: 2021 IEEE 37th International Conference on Data Engineering, ICDE, IEEE, 2021, pp. 1829–1834.

[32] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber, Bigtable: A distributed storage system for structured data, ACM Trans. Comput. Syst. (TOCS) 26 (2) (2008) 1–26.

[33] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis, Dremel: interactive analysis of web-scale datasets, Proc. VLDB Endow. 3 (1–2) (2010) 330–339.

[34] Jeff Shute, Radek Vingralek, Bart Samwel, Ben Handy, Chad Whipkey, Eric Rollins, Mircea Oancea, Kyle Littlefield, David Menestrina, Stephan Ellner, et al., F1: A distributed SQL database that scales, 2013.

[35] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al., Spanner: Google's globally distributed database, ACM Trans. Comput. Syst. (TOCS) 31 (3) (2013) 1–22.

[36] Ashish Gupta, Fan Yang, Jason Govig, Adam Kirsch, Kelvin Chan, Kevin Lai, Shuo Wu, Sandeep Dhoot, Abhilash Kumar, Ankur Agiwal, et al., Mesa: Geo-replicated, near real-time, scalable data warehousing, 2014.

[37] David F. Bacon, Nathan Bales, Nico Bruno, Brian F. Cooper, Adam Dickinson, Andrew Fikes, Campbell Fraser, Andrey Gubarev, Milind Joshi, Eugene Kogan, et al., Spanner: Becoming a SQL system, in: Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 331–343.

[38] Zhenkun Yang, Chuanhui Yang, Fusheng Han, Mingqiang Zhuang, Bing Yang, Zhifeng Yang, Xiaojun Cheng, Yuzhong Zhao, Wenhui Shi, Huafeng Xi, et al., OceanBase: a 707 million tpmc distributed relational database system, Proc. VLDB Endow. 15 (12) (2022) 3385–3397.

[39] Jianying Wang, Tongliang Li, Haoze Song, Xinjun Yang, Wenchao Zhou, Feifei Li, Baoyue Yan, Qianqian Wu, Yukun Liang, ChengJun Ying, et al., PolarDB-IMCI: A cloud-native HTAP database system at alibaba, Proc. ACM Manage. Data 1 (2) (2023) 1–25.

[40] Franz Färber, Norman May, Wolfgang Lehner, Philipp Große, Ingo Müller, Hannes Rauhe, Jonathan Dees, The SAP HANA database–An architecture overview, IEEE Data Eng. Bull. 35 (1) (2012) 28–33.

[41] Niloy Mukherjee, Shasank Chavan, Maria Colgan, Mike Gleeson, Xiaoming He, Allison Holloway, Jesse Kamp, Kartik Kulkarni, Tirthankar Lahiri, Juan Loaiza, et al., Fault-tolerant real-time analytics with distributed oracle database in-memory, in: 2016 IEEE 32nd International Conference on Data Engineering, ICDE, IEEE, 2016, pp. 1298–1309.

[42] Guoliang Li, Chao Zhang, HTAP databases: What is new and what is next, in: Proceedings of the 2022 International Conference on Management of Data, 2022, pp. 2483–2488.

[43] Mohammad Alomari, Michael Cahill, Alan Fekete, Uwe Rohm, The cost of serializability on platforms that use snapshot isolation, in: 2008 IEEE 24th International Conference on Data Engineering, IEEE, 2008, pp. 576–585.

[44] TATP Benchmark Description (Version 1.0), 2009.

[45] Patrick E. O'Neil, Elizabeth J. O'Neil, Xuedong Chen, The star schema benchmark (SSB), Pat 200 (2007) 50.

[46] Hasso Plattner, A common database approach for OLTP and OLAP using an in-memory column database, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, 2009, pp. 1–2.