OLxPBench: Real-time, Semantically Consistent, and Domain-specific are Essential in Benchmarking, Designing, and Implementing HTAP Systems

Guoxin Kang[1,2], Lei Wang[1,3], Wanling Gao[1,3], Fei Tang[1,2], and Jianfeng Zhan[1,2,3]
[1] Research Center for Advanced Computer Systems,
Institute of Computing Technology, Chinese Academy of Sciences
{gaowanling, wanglei_2011, zhanjianfeng}@ict.ac.cn
[2] University of Chinese Academy of Sciences
{kangguoxin, tangfei}@ict.ac.cn
[3] International Open Benchmark Council

***Abstract*—As real-time analysis on the fresh data become increasingly compelling, more organizations deploy Hybrid Transactional/Analytical Processing (HTAP) systems to support real-time queries on data recently generated by online transaction processing. This paper argues that real-time queries, semantically consistent schema, and domain-specific workloads are essential in benchmarking, designing, and implementing HTAP systems. However, most state-of-the-art and state-of-the-practice benchmarks ignore those critical factors. Hence, at best, they are incommensurable and, at worst, misleading in benchmarking, designing, and implementing HTAP systems. This paper presents OLxPBench, a composite HTAP benchmark suite. OLxPBench proposes: (1) the abstraction of a hybrid transaction, performing a real-time query in-between an online transaction, to model widely-observed behavior pattern – making a quick decision while consulting real-time analysis; (2) a semantically consistent schema to express the relationships between OLTP and OLAP schema; (3) the combination of domain-specific and general benchmarks to characterize diverse application scenarios with varying resource demands. Our evaluations justify the three design decisions of OLxPBench and pinpoint the bottlenecks of two mainstream distributed HTAP DBMSs. International Open Benchmark Council (BenchCouncil) sets up the OLxPBench homepage at https://www.benchcouncil.org/olxpbench/. Its source code is available from https://github.com/BenchCouncil/olxpbench.git.**

***Index Terms*—HTAP, benchmark, real-time analysis, semantically consistent schema, domain-specific workload.**

I. Introduction

In recent years, hybrid transaction/analytical processing (in short, HTAP) systems are proliferating fast. Many giant companies provide HTAP systems, including Oracle database in-memory [23], SQL Server [20], SAP HANA [33], MemSQL [14] and TiDB [4]. HTAP systems are popular for two reasons. First, giant companies are demanding fresher data in shorter shipping duration for real-time customer analysis [12]. Throughout this paper, real-time emphasizes performing a task like data analysis or user behavior simulation interactively in contexts like financial fraud monitoring or recommendation [47] [48]. Our usage of real-time is different from its traditional definition that limits the operations to completion within a hard or soft deadline [8]. The value of mass business data will diminish with time [39]. Moving data from the OLTP system to the OLAP system is complex and time-consuming. Meanwhile, it is impossible to perform real-time analysis on data that has passed a long turnaround time. Besides, software development and maintenance for two separate systems are also expensive. Second, the advanced modern software and hardware technologies, including in-memory computing (IMC) techniques [13], multi-core processors, various levels of memory caches, and large memories [25] [26], contribute to the HTAP systems' rapid development.

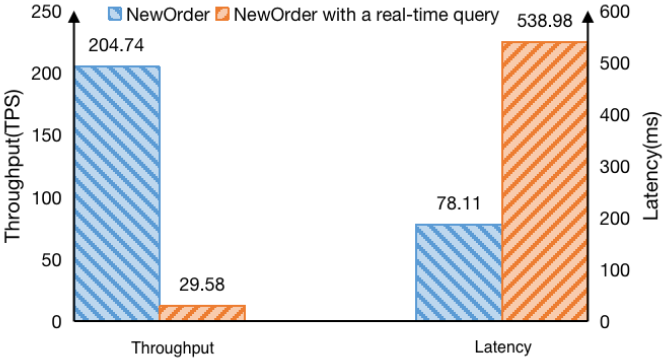


Fig. 1. This figure reveals the impact of a hybrid workload – performing a real-time query in-between an online transaction on the performance of TiDB – a state-of-the-art HTAP system against that of only an online transaction. The significant performance gap justifies why we should consider real-time queries in benchmarking HTAP systems. The other two critical factors that the HTAP benchmarks must consider are semantically consistent schema and domain-specific workloads.

There are three primary architectures for HTAP systems design and implementation. The first one introduces extract-transform-load (ETL) processing [27] between the OLTP DBMS and the data warehouse to complete data migration, data format transformation, and complex data analysis. However, the ETL systems are not competent for real-time analysis as they introduce time and space costs that can not

‡Jianfeng Zhan is the corresponding author.

be neglected. The second one is to utilize a stream processing system [21] [29] that feeds the incoming data to a batch processing system [30] [31]. Due to several unfavorable factors such as strong consistency semantics and double operating cost, it is not easy to use separate stream processing systems as HTAP solutions. The other architecture uses a single HTAP DBMS [4], [24], [32], [33], achieving high performance for OLTP and OLAP. It eliminates data movement overhead but adds pressure to performance isolation or consistency guarantees. In the face of numerous HTAP solutions, it is hard to compare different system performances. We describe these three architectures in detail in Section III-A.

The HTAP benchmarks provide quantitative metrics, methodology, and tools to evaluate different systems and their specific design decisions, offering valuable design and implementation inputs. However, as summarized in Table I, most state-of-the-art [41] and state-of-the-practice [40] HTAP benchmarks fail to consider real-time, semantically consistent, and domain-specific [37] in benchmarking, designing, and implementing HTAP systems. Hence, at best, they are incommensurable and, at worst, misleading in benchmarking, designing, and implementing HTAP systems. We quantitatively justify why we consider those three critical factors.

First, being real-time is essential. On the one hand, real-time is crucial to customer analysis – the fresher the data, the higher the value. On the other hand, there are widely-observed user behavior patterns – performing real-time analysis before making a quick decision. For example, if the customer wants to order an item in e-commerce, a query to get the lowest price rather than the random price of the item is most likely to happen before ordering the item. We propose the abstraction of a hybrid transaction, which performs a real-time query in-between an online transaction, to model this user behavior pattern, which the previous HTAP benchmarks overlook. Figure 1 shows the impact of a hybrid transaction on the performance of the online transactions in TiDB [4] – a state-of-the-art HTAP system against that of a sole online transaction as a baseline. The real-time query increases the baseline latency by a factor of 5.9, and decreases the baseline throughput by a factor of 5.9.

Second, the previous works use stitch schema. For example, the stitch schema in state-of-the-art HTAP benchmarks [40] [41] just reuses the schema from TPC-C [7] and TPC-H [9]. Instead, in real-world application scenarios, the data operated by the analytical query is generated by the online transaction, so the OLAP schema is a subset of the OLTP schema. We call this characteristic semantically consistent. The stitch schema can not disclose the severe interferences between analytical workload and transactional workloads in real-world scenarios. Our experiment result shows that the analytical workloads decrease transactional throughput by 89% using the semantically consistent schema, rather than 10% using stitch schema in the previous work [4].

Last but not least, most of the previous works only provide a general HTAP benchmark. The generic benchmark reflects a wide range of use cases. Instead, real-world applications have different workloads and schema with varying resource demands. So we propose the domain-specific benchmarks, which are specialized to evaluate the performance of HTAP systems in one or several specific scenarios. In Section VI, our evaluation shows that the peak transactional throughput of a general benchmark is nearly 20 times that of a domain-specific benchmark on the same testbed. Their performance varies greatly depending on the complexity of the relationships in the table, the read/write ratio of the transaction, and different system resource requirements.

We propose an HTAP benchmark suite in addition to a benchmarking framework named OLxPBench to help users perform performance comparisons. The main contributions of this paper are as follows. (1) We quantitatively justify why we should consider real-time queries, semantically consistent schema, and domain-specific workloads in HTAP benchmarking [2]. OLxPBench proposes new built-in hybrid workloads that perform a real-time query in-between an online transaction; a semantically consistent HTAP schema; one general benchmark; and two domain-specific benchmarks to evaluate the HTAP systems, including retail, banking, and telecommunications [12].

(2) We design and implement an extensible HTAP benchmarking framework and three comprehensive HTAP benchmarks: subenchmark, fibenchmark, and tabenchmark; Compared against the most related work – CH-benCHmark [40], our work is not trivial because there are eighteen analytical queries and seventeen hybrid queries tailored for different benchmarks. OLxPBench provides valuable experience in designing and implementing schematically consistent schema, hybrid workloads, and domain-specific benchmarks in HTAP benchmarking.

(3) Extensive experiments are conducted on the two mainstream HTAP systems: MemSQL and TiDB using OLxPBench against CH-benCHmark [40]. We have observed the following insights. The vertical table technique adopted by MemSQL is not very helpful to deal with the hybrid workloads because a large number of join operations generated by relationship query statements increases the waiting time of the hybrid transactions; The mainstream HTAP systems have poor performance in scanning operations for composite primary keys; The mutual interference between online transactions and analytical queries causes poor performance isolation.

II. Related Works

We compare OLxPBench with five other state-of-the-art and state-of-the-practice HTAP benchmarks in Table I. We classify the HTAP benchmarks into two groups according to the complexity of their workloads. The one contains intricate transactions and queries, such as CH-benCHmark [40], CBTR [25], and HTAPBench [41]. The other includes a mix of simple insert/select operations, i.e., ADAPT [42] and HAP [43]. The real-time queries generally involve simple *aggregate* operations and the analytical queries include more complex operations.

TABLE I
Comparison of OLxPBench With State-of-the-art and State-of-the-practice Benchmarks.

Name	Online transaction	Analytical query	Hybrid transaction	Real-time query	Semantically consistent schema	General benchmark	Domain-specific benchmark
CH-benCHmark	√	√	×	×	×	√	×
CBTR	√	√	×	×	√	×	√
HTAPBench	√	√	×	×	×	√	×
ADAPT	×	×	×	×	√	√	×
HAP	×	×	×	×	√	√	×
OLxPBench	√	√	√	√	√	√	√

CH-benCHmark launches the online transactions adopted from TPC-C and analytical queries from TPC-H concurrently on the stitch schema; however, they have different business semantics. Moreover, CH-benCHmark never updates the Supplier, Nation, and Region tables used by OLAP since the online transactions only update partial OLAP tables using stitch schema. Thus, its OLTP and OLAP operate different data and further cover up the contention in massive concurrency. HTAPBench uses the same schema model with CH-benCHmark. Besides, HTAPBench [41] implements the Client Balancer to control the number of analytical queries to avoid too many analytical queries affecting the performance of online transactions. CBTR mimics the order-to-cash process of real enterprise systems [25] and provides more complex as well as dynamic workloads. It is indisputable that CBTR has effective instruction for ad-hoc database design. Unfortunately, CBTR does not include the real-time query and is not open-source. Besides, its single domain-specific benchmark is insufficient to evaluate HTAP solutions in various scenarios.

ADAPT benchmark has two tables – a narrow table and a wide table [42]. The operations are abstracted from a natural productive environment, and the read-only operations are 80%. The HAP benchmark is based on the ADAPT benchmark and expands the update and deleted operation to test the storage engine. The read-only operations are 50%. Overall, the operations are too simple to represent complex transactions in natural business environments.

When designing OLxPBench, we choose the semantically consistent schema rather than the stitched schema [41] [40] to expose the original interference between OLTP workloads and OLAP workloads. Besides, we increase the real-time query for real-time user behavior analyzing and simulating. We also obey both general [7] and domain-specific [22] principles. The general benchmark helps designers perform performance comparisons, and the domain-specific benchmarks help the user select the HTAP DBMS that best support their specific workloads. Moreover, we compare the semantically consistent schema to stitched schema [40] in Section V-B1 because CH-benCHmark [40] originates from the HTAP transactional benchmark.

III. Background and Motivation

A. The Background of HTAP Systems

HTAP DBMSs need to perform trade-offs considering different performance requirements of different workloads. Currently, there are three types of HTAP solutions, and we compare their pros and cons in the following subsections.

The first solution is to use separate DBMSs [16], [19], [44], [45] to achieve high performance of online transactions and analytical queries. Generally, online transactions adopt a row-based store due to its high efficiency for records insert and update. Analytical queries often adopt a column-based data warehouse since it supports efficient data scans. However, separate DBMS needs to convert row-based data to column-based ones, i.e., ETL processing, which is too time-consuming to analyze the latest data and make instant decisions. For example, Pavlo et al. [39] [26] refer to the standard ETL process to migrate data from the OLTP system to the OLAP system as one of the solutions to HTAP.

Second, the lambda architecture [28], [34], [36], which consists of a real-time stream processing system and a batch processing system, can perform real-time analytics on incoming data, but it is expensive. The real-time stream processing [21] systems provide views of online data while simultaneously using batch processing [6] to provide comprehensive and accurate views of batch data. Besides, the serving layer merges the real-time view with the batch views and then responses to the end-user. The lambda architecture provides a real-time analysis at a considerable cost, including double write costs, double or more development costs, and so on. In brief, the cost of maintaining two systems is also very high.

Third, using a single HTAP DBMS to handle online transactions and real-time queries. Because real-time analytics on fresh data is valuable, the lightweight propagation technique is developed to transfer recent transactional logs to the analytical storage node in a more short and flexible schedule [24]. Microsoft SQL Server [20] stores hot inserted and updated rows in middle delta storage to transfer them to OLAP storage and speed up query processing. Oracle in-memory database [23] keeps a dual-format store for OLTP and OLAP workloads without double memory requirements. It uses a read-only snapshot maintained in memory for analysis. Their latest work [35] provides a more available distributed architecture and fault tolerance than the original. MemSQL uses an in-memory row-store and an on-disk column-store to handle highly concurrent operational and analytical workloads [14]. The storage layer of TiDB consists of a row-based store and a column-based store. To analyze real-time queries of the fresh data, TiDB uses asynchronous log replication to keep the data consistent [4]. The IBM Db2 Analytics Accelerator uses replication technology to enhance its real-time capacity [17]. VEGITO [46] retrofits the high availability mechanism to support HTAP workloads.

B. Motivation

1) It is mandatory to include real-time queries in HTAP benchmarking: HTAP benchmarking should contain real-time queries for the following reasons. First, real-time queries matter in customer analysis. HTAP DBMSs enable more informed and in-business real-time decision-making [11]. Real-time customer analysis is crucial because it is the basis of instant decision-making and fraud detection. The fresher the data, the higher the value. Real-time queries are usually executed on the recent data committed by transactions. For example, if an item requested by a customer has been sold out according to the real-time inventory, similar ones will be recommended instantly.

Second, real-time queries can be used to mimic real-time user behavior. For example, if a customer wants to create a New_Order transaction in TPC-C [7], what is most likely to happen before selecting an item during the New_Order transaction [40] is a real-time query that finds the lowest price of the goods, rather than the random price. However, none state-of-the-art [41] and state-of-the-practice [40] HTAP benchmarks provide the workloads that include real-time queries imitating user behavior.

Figure 1 shows the impact of a real-time query on the performance of TiDB [4] – a state-of-the-art HTAP system. The New_Order transaction is the same as the New_Order transaction in TPC-C [7]. The real-time query is an aggregate operation that gets the item's lowest price in real time. Real-time queries in the subenchmark are from a top-tier E-commerce internet service provider. The experimental setup is the same as Section V-A. When a real-time query is injected in the New_Order transaction [41] [40], the average latency increases by 5.9x, and the throughput reduces by 5.9x. So it is mandatory to include real-time queries in the HTAP benchmarks, or else the evaluation result will be misleading [2].

2) Semantically consistent schema is essential: The state-of-the-art [41] and state-of-the-practice [40] HTAP benchmarks all use stitch schema. The stitch schema integrates the TPC-C schema [7] with TPC-H [9] schema and includes 12 tables. The $NEW-ORDER$, $STOCK$, $CUSTOMER$, $ORDERLINE$, $ORDERS$, and $ITEM$ tables are accessed by TPC-C and TPC-H. The $WAREHOUSE$, $DISTRICT$, and $HISTORY$ tables are only accessed by TPC-C. The $SUPPLIER$, $NATION$, and $REGION$ tables are accessed by TPC-H only. Both TPC-C and TPC-H keep the third normal form to reduce data duplication.

There are two flaws in such a stitch schema: First, OLTP and OLAP operate on the same business data in real scenarios; however, the query only analyzes one-sided business data with the stitch schema, leading to biased decisions. For example, in CH-benCHmark, the stitch schema only allows queries to analyze data from the shared six tables between TPC-C and TPC-H. When the $Payment$ transaction in CH-benCHmark is completed, a record will be written in the history table. The records in the history table are essential for analyzing the custom's behavior. However, none of the analysis queries in previous benchmarks [41] [40] can analyze the tens of thousands of records in the history table. In addition, there is no query to analyze the warehouse table and district table of TPC-C [41] [40]. It is very costly to discard valuable parts of OLTP data. The stitch schema leads the results of the analytical queries to be partial, perplexing, and incorrect.

Second, with stitch schema, the competitions between analytical workload and transactional workloads are hidden, making it impossible to fairly evaluate the interference between analytical workload and transactional workloads in real-world scenarios. The online transactions and analytical queries operate on the same business data in the real world, so intense competitions for resources are not avoidable. However, in the previous benchmark [40], [41], 45.4%, 40.9%, and 13.6% of the 22 queries on the stitch schema access the $SUPPLIER$, $NATION$, and $REGION$ tables that never update or insert records, respectively. The low competition between analytical and transactional workload will propagate a false image that the HTAP system can guarantee the isolated performance for separate OLTP and OLAP workloads [4]. In Section VI-A2, we use the general benchmark in OLxPBench, which uncovers the high competition between the OLTP and OLAP workloads, to evaluate the TiDB and find the throughput interference of OLTP and OLAP is as high as 89% and 59%, respectively.

3) Domain-specific benchmarks should be included: CH-benCHmark [40] is a general benchmark that fails to evaluate the HTAP system performance in a particular application scenario. In Section VI, our evaluation shows that the peak transactional throughput of a general benchmark is nearly 20 times that of a domain-specific benchmark on the same testbed. In the face of numerous HTAP solutions, there is an urgent need to consider generic and domain-specific HTAP DBMS benchmarks. OLxPBench provides one generic benchmark and two domain-specific benchmarks for evaluating HTAP systems. In Sections VI-B1 and VI-C1, our evaluation shows that the peak transactional throughput of a domain-specific benchmark is nearly 200 times that of another domain-specific benchmark on the same testbed. OLxPBench implements an extensible framework that makes it easy for developers to add a new benchmark.

IV. The design and implementation

To fully evaluate HTAP DBMSs, we present OLxPBench, consisting of a general benchmark and two domain-specific benchmarks. The general benchmark, which we name subenchmark, extracts complex operations from the retail activity and does not attempt to model an actual application scenario [7], which intends to perform performance comparison for HTAP DBMSs. Meanwhile, OLxPBench has two domain-specific benchmarks, which we name fibenchmark and tabenchmark, model the financial [1] and telecommunication [10] scenarios, and help the users select the HTAP DBMS that best support their specific workloads. This section introduces the OLxPBench suite from the schema model design, the details of workloads, the benchmark categories, and implementation.

TABLE II
FEATURES OF THE OLxPBENCH WORKLOADS.

Benchmark	Tables	Columns	Indexes	OLTP Transactions	Read-only OLTP Transactions	Queries	Hybrid Transactions	Read-only Hybrid Transactions
Subenchmark	9	92	3	5	8.0%	9	5	60.0%
Fibenchmark	3	6	4	6	15.0%	4	6	20.0%
Tabenchmark	4	51	5	7	80.0%	5	6	40.0%

A. HTAP Schema model design

We follow three principles in designing the HTAP schema.

(1) Any record accessible to OLTP should be accessible to OLAP. Because online transactions generate the data that the analytical queries will analyze. The OLTP schema set should include the OLAP schema. We first propose that in the HTAP benchmarks, the mixed workloads, including OLTP and OLAP workloads, should use the semantically consistent schema. They will reveal the inherent interference between OLTP workload and OLAP workloads.

(2) The schema models should be diverse and practical for thoroughly evaluating the various HTAP solutions. The diversity of schema models is reflected in the diversity of practical uses. Therefore, we provide the generic schema model for performance comparisons and two domain-specific schema models for users to select the HTAP DBMS that best support their specific workloads. We choose schema models for retail, banking, and telecommunications activities because the above practitioners were among the first to adopt the HTAP solutions [12].

(3) The design of integrity constraints should be relevant to implementing a specific HTAP database. For example, some HTAP DBMSs (such as MemSQL [14]) do not currently support foreign keys. As a result, OLxPBench's schema models come in two versions, one with no fundamental foreign constraint and one with foreign constraint, which users can choose on-demand.

B. The details of HTAP Workloads

OLxPBench contains nine built-in workloads with different types and complexity. Three online transaction workloads are extracted from popular benchmarks [10] [7] [38]. In addition, we add three analytical query workloads and three hybrid transaction workloads for real-time customer analysis and simulating the real-time user behaviors. We distill the E-commerce services from an industry partner, which we keep anonymously at its request, into representative real-time queries. The analytical workloads contain complex analytical operations such as $multi{-}join$, $sub{-}selection$, $Group{-}By$, and $Order - By$ operations based on the different schema models. Table II describes the features of these benchmarks.

In more specific implementations, we modify the integrity constraints of the schema of SmallBank [1] and TATP [10] to adapt the implementation of the MemSQL [14]. Furthermore, we increase the composite primary key to TATP [10], which is common in real business scenarios. OLxPBench provides valuable experience for schema model design and hybrid transaction abstraction. The request rates, transaction/query weights, and schema relations are configurable for different testing purposes. This subsection will introduce the details of these benchmarks in turn.

1) Subenchmark: The subenchmark is inspired by TPC-C [7], which is not bound to a specific scenario, and the community considers a general benchmark for OLTP system evaluation. The online workloads of the subenchmark are the same as TPC-C's transactions, which are write-heavy and merely 8% read-only transactions. The online transactions include $NewOrder$, $Payment$, $OrderStatus$, $Delivery$, and $StockLevel$.

The nine analytical queries in the subenchmark keep the essential characteristics such as the complexity of operations. The analytical queries perform multi-join, aggregation, grouping, and sorting operations on a semantically consistent schema. For example, the Orders Analytical Report Query (Q1) is designed for getting the magnitude summary for all *ORDER_LINE* items as of a given date. The query lists the total quantity, total amount, average quantity, and average amount for further analysis. The above aggregates are grouped by their number and listed in ascending order. We newly increase five hybrid transactions, and the default configuration of the subenchmark has 60% read-only hybrid transactions. The real-time queries that simulate the user behavior are the representative aggregation operations in the actual E-commerce production application: if the customer wants to create a New_Order transaction, a query to get the lowest price rather than the random price of the item (X1).

2) Fibenchmark: The fibenchmark is inspired by SmallBank [1], which aims at bank scenarios. Hence, it is a domain-specific benchmark. The fibenchmark contains three tables: *ACCOUNT*, *SAVING*, and *CHECKING*, and the transactions mainly modify the customers' accounts. The online transactions are *Amalgamate*, *Balance*, *DepositChecking*, *SendPayment*, *TransactSavings*, and *WriteCheck*. Fifteen percent of the above transactions are real-only in the default configuration. We keep all the online transactions of SmallBank [1], and we newly increase the analytical workloads and the hybrid transactions in fibenchmark.

The analytical workloads perform real-time customer account analytics. The complex queries include $join$, $aggregate$, $sub - selection$, $Order - By$ and $Group - By$ operations. For example, the Account Name Query (Q1) lists the name in the combining row from *ACCOUNT* and *CHECKING* tables. Besides, the real-time queries in hybrid transactions are generally the $aggregate$ operations and perform the real-time financial analysis on the user's account. There are six hybrid transactions, and the default configuration of the fibenchmark has 20% read-only hybrid transactions. For example, the Checking Balance Transactions (X6) checks whether the cheque balance is sufficient and aggregates the

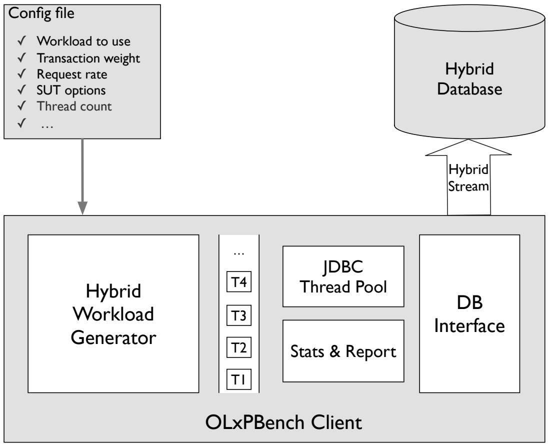


Fig. 2. OLxPBench architecture.

value of the minimum savings. The volatility of extreme values is also an important research topic in the financial field.

3) Tabenchmark: The tabenchmark is inspired by TATP [10], which aims at telecom scenarios. Hence, it is a domain-specific benchmark. The online transactions in the tabenchmark simulate a typical Home Location Register (HLR) database used by a mobile carrier [10]. Eighty percent of online transactions are real-only and the transactions are *DeleteCallForwarding*, *GetAccessData*, *GetNewDestination*, *GetSubscriberData*, *InsertCallForwarding*, *UpdateLocation* and *UpdateLocation*. We modify the primary key of the $SUBSCRIBER$ table from s_id to (s_id, sf_type), because the composite primary key is standard in the real business scenario. The original data definition language file is a choice. We keep all the online transactions of TATP [10], and we newly increase the analytical workloads and the hybrid transactions in fibenchmark.

The analytical queries help the mobile network operators to analyze user behavior in real-time. The analytical queries also comprise the $arithmetic$ operation besides the operations in the fibenchmark. For example, the Start Time Query (Q3) calculates the average of the starting time of the call forwarding. The average value of start time is essential for load forecasting. The real-time queries perform the real-time activities on practical mobile users. The real-time query not only performs aggregation operation but also does a fuzzy search based on the sub-string. For example, the Fuzzy Search Transaction (X6) queries all information about the subscriber. It selects the subscriber IDS whose user data matches the fuzzy search criteria.

C. The implementation of OLxPBench

OLxPBench is used for evaluating distributed HTAP DBMSs and other HTAP DBaaS systems that support SQL through JDBC. The architecture of the OLxPBench is shown in Figure 2. OLxPBench parses configuration files at runtime and generates the corresponding hybrid workloads. Then the hybrid workloads are populated in request queues. The request rates, transaction types, real-time query types, weights, and target DB configuration are specified in the XML file. The thread connects to the target database by JDBC pool and pulls requests from the request queue. The threads also measure the latency and throughput metrics. Finally, the statistics module aggregates the above metrics and stores the min, max, medium, 90th, 95th, 99.9th, and 99.99th percentile latency in a file specified by the user in the terminal.

The open-loop mode sends the requests with the precise request rate control mechanism because the open-loop load generator sends the request without waiting for the previous request to come back. However, in a closed-loop mode, the response of a request triggers the sending of a new request. Besides, the users can customize the weights of various online transactions and analytical queries. OLxPBench is inspired by the OLTP-Bench's OLTP module [18] and newly increased analytic and hybrid modules. OLxPBench achieves three online and analytical agent combination modes for different HTAP solutions. The first mode sequentially sends the online transactions or analytical queries. The second mode concurrently invokes the transactional workload and analytical workload. The last mode sends hybrid transactions performing a real-time query in-between an online transaction to simulate the user behavior. The OLxPBench client is a java program and is easy to extend with new hybrid database back-ends.

V. Evaluation

Our evaluation illustrates the effectiveness of OLxPBench. In Section V-B, we compare OLxPBench with the state-of-the-practice [40] work, testing the key features of OLxPBench and reporting the standard deviation of absolute value. From Section VI-A to Section VI-D, we evaluate the mainstream distributed HTAP DBMSs using OLxPBench and pinpoint the bottlenecks of two mainstream distributed HTAP DBMSs. In Section VI-E, we evaluate the scaling capability of TiDB, MemSQL, and OceanBase.

A. Experimental Setup

1) Cluster Deployment: We deploy a 4-node cluster for our evaluation. Each server includes 2 Intel Xeon E5-2620@2.40GHz CPUs, 64 GB memory, and 1TB SSD. Each CPU has 6 physical cores, and hyper-thread is enabled. We used all of the 24 hardware threads. For the scaling capability experiments in Section VI-E, we deploy a 16-node cluster, with each cloud server including 8 Intel Xeon Platinum 8269CY@2.50GHz virtual CPUs, 32 GiB memory, and 140GiB enhanced solid-state disk (ESSD). We used all of the 8 threads. All machines are configured with Intel Ethernet 1GbE NICs. The operating system is ubuntu 16.04 with the Linux kernel 4.4.

2) Database Deployment: In our experiments, 4-node configuration ensures that the components of systems are under test are distributed deployed. TiDB [4] is a Raft-based HTAP database. TiSpark is a powerful analysis engine to help TiDB connect to the Hadoop ecosystem. The SQL engine processes process online transactions and analytical queries.

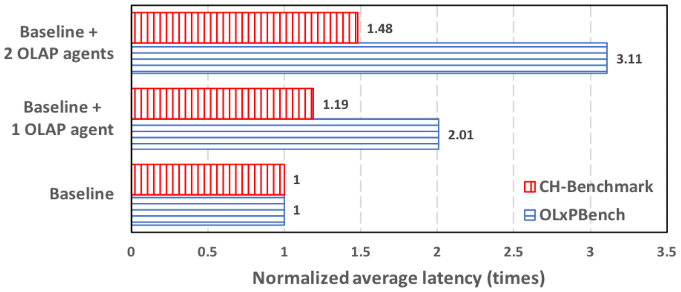


Fig. 3. Comparing the schema model of OLxPBench and CH-benCHmark on TiDB cluster.

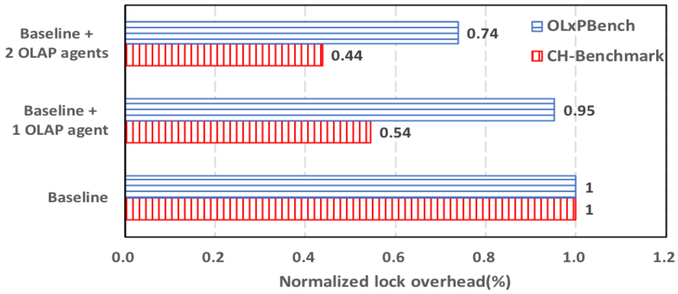


Fig. 4. Comparing the lock overhead of different schema models.

The distributed storage layer consists of a row store (TiKV) and a columnar store (TiFlash). Two TiKV instances are deployed on two servers with a TiDB SQL engine instance. Two TiFlash instances are deployed on the other servers with a TiSpark instance. The TiDB [4] version is 5.0.0-RC, and the number of replication is two. TiDB provides snapshot isolation or repeatable read isolation levels, and we choose repeatable read isolation in the experiments. The MemSQL [14] cluster consists of aggregator nodes and leaf nodes. The aggregator nodes receive queries and distribute them to leaf nodes. The leaf nodes store data and execute queries issued by the aggregator nodes. The version of MemSQL is 7.3, and the number of replication is two. A MemSQL cluster consists of at least one master aggregator node and one leaf node. We keep two leaf nodes, one aggregator node, one master aggregator node in 4 separate servers. MemSQL only supports a read committed isolation level. We remain the default configurations of the above two distributed hybrid databases.

3) Workloads Deployment: The workloads in the following experiments have three sources: subenchmark, fibenchmark, and tabenchmark. Each benchmark contributes two composites of workloads: (1) the OLTP agents and OLAP agents launch the mixtures of online transactions with analytical queries; (2) The hybrid agents send the hybrid workloads performing a real-time query in-between an online transaction to simulate the user behavior. We do not test the performance of the cold start procedures, so there is a 60-second warm-up period before the 240-second run time. All the workloads are open-loop, and the request rates used in the experiment depend on the cluster's peak performance. The requested rates vary during the experiment, and the warehouse quantity is 50. The interference between online transactions and analytical queries increases with the increasing request rates. The transactional/analytical request rate unit is transactions per second (tps). OLxPBench reports the average latency, tail latency, and throughput.

B. The evaluation of OLxPBench key design features

In this subsection, we demonstrate the design features of OLxPBench. **(1) Which schema model to adopt**, **(2) how the real-time query impacts the performance of HTAP systems**, and **(3) Why the domain-specific benchmark should be considered**. The following results are the average results of the three runs.

1) Schema Model: We explore the performance difference between the semantically consistent and traditional stitched schema in the same TiDB [4] cluster. We choose the state-of-the-practice HTAP benchmark, CH-benCHmark [40], as the reference. Because CH-benCHmark [40] is still the most popular HTAP benchmark. The average number of requests L equals the long-term average arrival rate λ multiplied by the average latency W that a request spends in the system. The Little's Law [3] is

$$L = \lambda W \tag{1}$$

According to Little's Law [3], the load stress in the TiDB is directly influenced by the average number of requests L rather than the different average request arrival rates λ of open data generator (OLxPBench) and closed data generator (CH-benCHmark). So, when the average number of requests L in the queuing system (TiDB) is fixed, the load stress in the TiDB is fixed. The average number of online transactions in a stable TiDB cluster over the long term is around 45. We drop the write-heavy transactions such as *NewOrder* and *Payment* to reduce the possibility of load imbalance.

Test Case 1: Varied OLAP pressures. The sum of the OLAP thread increases from zero to two. We put the incremental OLAP pressure on the tested systems to disturb the performance of online transactions and compare the latency of the different schema designs. We normalized the baseline of OLxPBench and CH-benCHmark to make a fair comparison. Little's Law does not guarantee that OLxPBench and CH-benCHmark have the same transmission rate of request, so the absolute value comparison is unfair. Figure 3 shows that the normalized average latency of online transactions in OLxPBench is more than double with the lowest OLAP pressure compared to without the OLAP pressure. However, the normalized average latency of online transactions in CH-benCHmark increases by no more than one-fifth of the baseline under the same OLAP pressure. Under the enormous OLAP pressure, the normalized average latency of online transactions in OLxPBench increases more than three times. Each OLAP thread sends one OLAP query per second. The OLAP queries are time-consuming scan tables operations that bring a large amount of data into the buffer pool and evict an equivalent amount of older data. Two OLAP threads can generate enormous pressure and cause a significant increase in server-side average CPU utilization. At the same time, the normalized average latency of online transactions in CH-benCHmark increases by around 48 percent of the baseline.

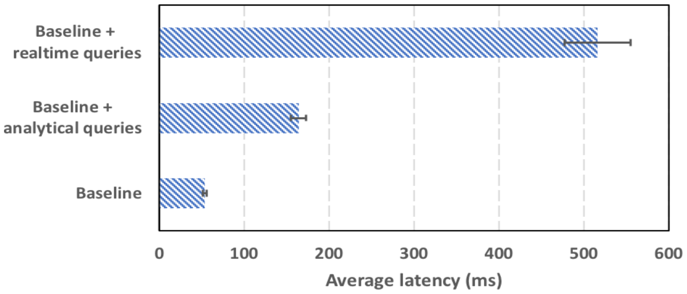


Fig. 5. Comparing the analytical queries to the real-time queries of subenchmark on the TiDB cluster.

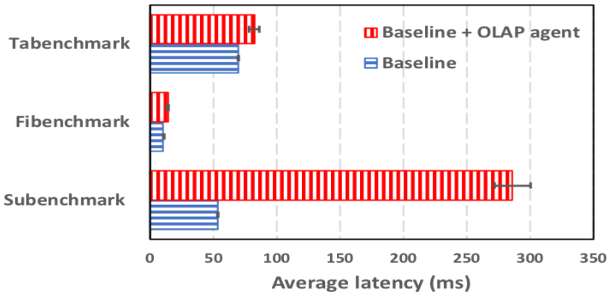


Fig. 6. Comparing the generic benchmark to the domain-specific benchmarks on TiDB cluster.

TiDB [4] provides a row-based store, TiKV, and a column-based store, TiFlash. The data in TiFlash keep consistent with the data in TiKV using asynchronous log replication. Nevertheless, the scan tables operations can occur in the row store of TiKV or the column store of TiFlash. As the number of OLAP agents increases, the number of scan table operations increases. We use the performance monitoring unit tool such as Perf to obtain the performance events and count the overhead of the lock. According to the Linux Perf tool's manual, samples are performance data collected using the ' perf record' command. Lock samples indicate the number of samples collected in the lock function. Lock overhead includes the syscall overhead of mutual exclusion (mutex) locks, fast userspace mutex (futex), and spinlock. Lock overhead *LO* equals the number of lock samples *LS* divided by the total number of samples *TS*. The baseline lock overhead *BLO* is the lock overhead of the online transactions without analytical query influencing. The normalized lock overhead *NLO* is the lock overhead *LO* divided by the baseline lock overhead *BLO*.

$$NLO = \frac{LS}{TS * BLO} \times 100\% \tag{2}$$

When the analytical agent increases, the throughputs of online transactions will be influenced. So the normalized lock overhead decreases with the analytical agent increases in Figure 4. The difference in performance isolation measured by OLxPBench is far more significant than CH-benCHmark. Better performance isolation indicates that the execution of OLTP with OLAP workloads affects the other one's performance much lighter. Figure 4 reports that the lock overhead gap between semantically consistent schema and stitched schema is 1.76x using one OLAP thread and 1.68x using two OLAP threads. It indicates that the shared data between OLTP and OLAP on a semantically consistent schema is more significant than stitched ones. The low competition in CH-benCHmark between OLTP and OLAP workloads will propagate a false image that the HTAP system can guarantee isolated performance.

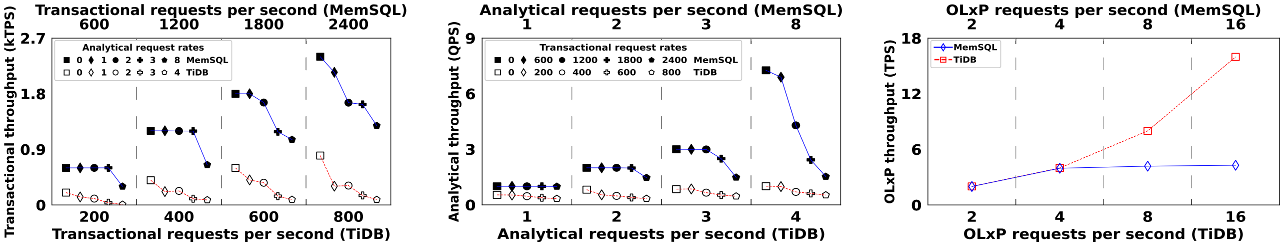
Implication 1 *Experiments show that semantically consistent schema reveals inherent competition between OLTP and OLAP than stitched schema.*

2) Real-time Query: We now compare the two main queries common in real-world scenarios: *analytical queries* and *real-time queries*. First, the analytical queries keep the essential characteristics such as the complexity of operations and perform multidimensional customer analysis. Second, the real-time queries in OLxPBench are extracted from the existing production applications. The real-time queries are used in the production applications to perform real-time user behavior simulations. However, none state-of-the-art [41] and state-of-the-practice [40] HTAP benchmarks provide the workloads that include real-time queries imitating user behavior. We compare the two different intentional queries on the TiDB cluster.

Test Case 2: Queries comparison. We run the subenchmark using the semantically consistent schema at 30 online transactions per second as the baseline. Then we inject analytical queries at 1 query per second into the baseline as experimental group one. Meanwhile, we send the hybrid transaction performing a real-time query in-between an online transaction at 30 requests per second as the experimental group two. Figure 5 shows that the analytical queries increase the baseline latency by around three times. The real-time queries in hybrid transactions increase the baseline latency by more than nine times. The hybrid transaction contains both OLTP statements and OLAP statements, but the SQL engine can only choose a row-based store or column-based store to handle the hybrid transaction. However, the analytical queries and the online transactions can be handled separately by the column-based TiFlash and the row-based TiKV. Therefore, the impact of real-time query simulating the user behavior is more significant than the impact of analytical queries on the performance of online transactions. Besides, the standard deviation of the average baseline latency is 2.21. With the analytical queries interference, the standard deviation of average baseline latency increases from 2.21 to 9.16. Under the real-time queries interference, the standard deviation of average baseline latency increases from 2.21 to 38.91. It indicates that interference of real-time queries to online transactions is greater than that of analytical queries. So it is necessary to include the real-time queries extracted from the production environment in the HTAP benchmark for helping users choose the appropriate HTAP system to handle real-time queries.

Implication 2 *It is necessary to include the real-time queries in the HTAP benchmark for testing whether the HTAP system can handle real-time queries from users.*

3) Domain-specific Benchmark: In this paper, we classify the benchmarks into two categories: *generic benchmark* and



(a) Throughput of OLTP.

(b) Throughput of OLAP.

(c) Throughput of OLxP.

Fig. 7. OLTP, OLAP and OLxP performance of subenchmark.

domain-specific benchmark. The subenchmark is inspired by TPC-C [7], which is not bound to a specific scenario, and the community considers a general benchmark. The fibenchmark and the tabenchmark model a banking scenario and a telecom scenario. Hence, they are domain-specific benchmarks.

Test Case 3: Domain-specific benchmark. We run the subenchmark, the fibenchmark, and the tabenchmark at 80 online transactions per second as the baseline. Then we send the analytical queries at 1 query per second with the baseline. Figure 6 shows that the baseline of the above three benchmarks is 53.47ms, 10.25ms, and 69.53ms. Moreover, the standard deviations of the baseline of the above three benchmarks are 0.23, 0.05, and 0.47. The online transactions of fibenchmark perform read-heavy and simple update operations, so its baseline latency is the smallest one. Slow queries took longer than one second in tabenchmark's online transactions. So the baseline of the tabenchmark is the biggest one. We will analyze the reason for slow queries in Section VI-E. Only 8% of online transactions in subenchmark do not modify the table, and the tables in subenchmark contain complex relations. So its baseline average latency is the median.

Under the OLAP pressure, the OLTP latency of subenchmark increases by more than five times, and the OLTP latency of fibenchmark increases by less than forty percent. And the OLTP latency of tabenchmark increases by less than twenty percent.

Meanwhile, the standard deviations of the above three benchmarks increase to 14.10, 0.58, and 4.05 under the analytical queries interference. The complex analytical queries in subenchmark generate many table scan operations, which increase the waiting time of online transactions. The read-heavy online transactions of fibenchmark are mostly negligible by OLAP agents. Therefore, online transactions of subenchmark are most affected by OLAP pressure, followed by fibenchmark' online transactions, and tabenchmark' online transactions are the least affected.

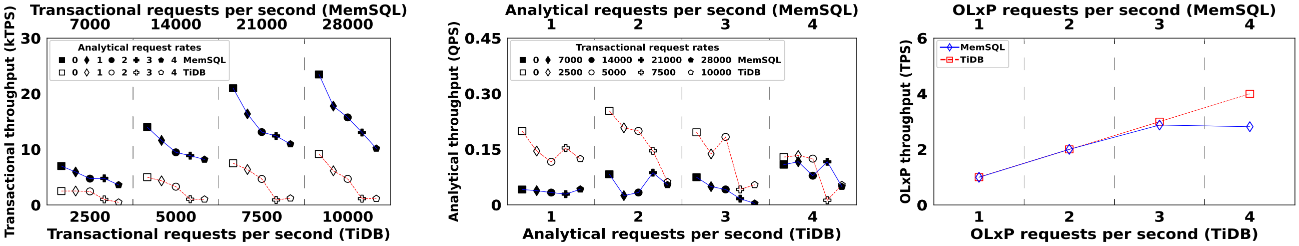
Implication 3 *The domain-specific benchmarks help users identify system bottlenecks in their specific scenarios. Besides, it also helps system designers point in the direction of system optimization.*

VI. Evaluation of The Mainstream Distributed HTAP DBMSs

In addition to the feature evaluation of OLxPBench, we also thoroughly test end-to-end performance for mainstream distributed HTAP DBMSs. We will provide detailed evaluation data in the following subsections, including peak performance. The peak performance refers to the saturation value that a single workload can reach in the test cluster. Furthermore, we will describe and deeply analyze the mutual interference between OLTP and OLAP [5] using the control variate method. The transactional/analytical request rates are divided into four numerically increasing groups with the same interval based on peak throughput. The transactional/analytical request rates in each group are the same, and the analytical/transactional request rates increase from zero to peak to explore the influence of the analytical/transactional agents on the transactional/analytical agents. Besides, CH-benCHmark [40] uses the stitch schema while OLxPBench uses a semantically consistent one. In addition, OLxPBench uses hybrid workloads. The difference in performance isolation measured by OLxPBench is far more significant than CH-benCHmark. Better performance isolation indicates the execution of OLTP with OLAP workloads affects the other one's performance much lighter. Moreover, we find that the lock overhead gap between the OLxPBench and CH-benCHmark is 1.76x under the same OLAP pressure in TiDB. The low competition in CH-benCHmark between OLTP and OLAP workloads will propagate a false image that the HTAP system can guarantee isolated performance. So, we do not report the experimental results of CH-benCHmark.

A. Subenchmark evaluation

1) Peak performance: Figure 7(a) shows that the transactional throughput increases with the incremental transactional request rates. In the MemSQL cluster, the throughput reaches the top when the transactional request rates are 2400 tps. The average latency of transactions is 29.7 milliseconds without OLAP agent inferences. And the 95th percentile latency of transactions is 78.53 milliseconds. In the TiDB cluster, the maximum transactional throughput is 800 tps. Figure 7(a) illustrates that the throttled transactional throughput of subenchmark in the TiDB cluster is one-third that of the MemSQL cluster. The above result is the data processing of MemSQL in memory rather than in solid-state disk. Figure 7(b) shows



(a) Throughput of OLTP.

(b) Throughput of OLAP.

(c) Throughput of OLxP.

Fig. 8. OLTP, OLAP and OLxP performance of fibenchmark.

that the maximum analytical throughput is around eight tps in the MemSQL cluster. Moreover, the analytical throughput reaches the top when the analytical request rates are four tps in the TiDB cluster. Under the same analytical request rates, the average latency of OLAP increases with the incremental transactional request rates.

The performance of OLxP in the different hybrid request rates is shown in Figure 7(c). The OLxP workloads include hybrid transactions, which perform a real-time query in-between an online transaction to simulate the user behavior. The real-time query is a time-consuming aggregate operation. So, the transactional statements behind the real-time query must wait for the real-time query execution because of the atomicity property of the transaction. So, the maximum throughput of OLxP is 4.28tps and 15.98tps in Figure 7(c). In the MemSQL cluster, the maximum average hybrid latency is 133.44 seconds. The gap between the maximum and minimum average delays is 223 times. And the 95th percentile latency of hybrid workload is 209.50 seconds. MemSQL adopts the vertical partitioning technology, which results in many join operations generated by relationship query statements in hybrid transactions and increases the waiting time of hybrid transactions. In the TiDB cluster, the throttled OLxP throughput is 16 tps, and the maximum average hybrid latency is 397 milliseconds. The maximum average delay is 1.47 times the minimum average delay. And the 95th percentile latency of hybrid workload is 905.36 milliseconds. The above results indicate that TiDB's separated storage engine can handle the OLxP workloads compared to MemSQL's single storage engine.

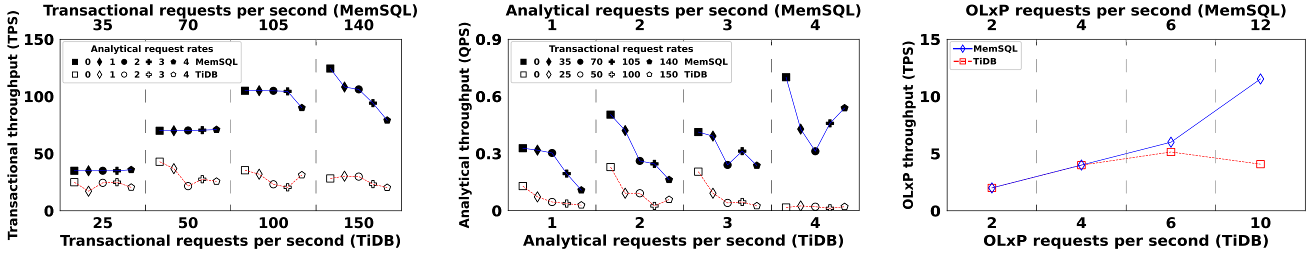
2) Performance interference between OLTP agents and OLAP agents: The performance impact of analytical agents on transactional agents is shown in Figure 7(a). When the transactional request rates are controlled, the average latency of the transactional agents increases by up to 17.4 times compared with the absence of the analytical agents in the MemSQL cluster. And the gap of the 95th percentile latency is 33.7x. The performance impact of the online transactions on the analytical queries is shown in Figure 7(b). When the analytical request rates are controlled, the average latency of the analytical agents increases by up to 2.2 times compared with the absence of the transactional agents. And the gap of the 95th percentile latency is 2.0x. It indicates that violent interference exists between transactional agents and analytical agents. The expensive analytical queries compete for the resource with the online transactions in the single storage and increase the latency of online transactions.

In the TiDB cluster, under the same analytical request rates, the analytical throughput decreases to the baseline of 59% as the transactional request rate increases. Furthermore, when the transactional request rates are 800 tps, the transactional throughput plummets as the analytical request rates increase, up to 89%. The transactional agents significantly affect the execution of analytical agents. The higher the request rates, the more table scan operations. Time-consuming table scan operations increase the waiting time of requests and reduce requests throughput.

B. Fibenchmark evaluation

1) Peak performance: As shown in Figure 8(a), the peak transactional throughput is around 23476 tps in the MemSQL cluster. The maximum transactional throughput is 9165 tps in the TiDB cluster. The read-only transaction ratio of fibenchmark is higher than that of subenchmark, so the peak transactional throughput of fibenchmark is higher than that of subenchmark. A large number of queries are blocked until the previous complex queries are completed. So the peak analytical throughput of MemSQL is around 0.12 tps in Figure 8(b). And the maximum analytical throughput of TiDB is 0.25 tps. There are a lot of scan table operations in the workloads of fibenchmark, and scanning row-format tables in TiKV [4] is stochastic and expensive, explained in Section V-B1. Figure 8(c) shows that the hybrid throughput increases as the hybrid request rates increase when the hybrid request rates are no more than four tps. In the MemSQL cluster, the peak hybrid throughput is 2.9 tps. In the TiDB cluster, the peak hybrid throughput is 4 tps. The average hybrid latency increases at most 17.2% as the hybrid request rates increase. And the 95th percentile latency increases up to 36.4% as the hybrid request rates increase. The hybrid latency increase as the hybrid request rates increase without bound. The increasing average and 95th percentile latency of hybrid transactions result from more waiting time with the higher hybrid request rates.

2) Performance interference between OLTP agents and OLAP agents: The performance impact of analytical agents on transactional agents is shown in Figure 8(a). And the performance impact of transactional agents on analytical agents is shown in Figure 8(b). In the MemSQL cluster, the trans-



(a) Throughput of OLTP.

(b) Throughput of OLAP.

(c) Throughput of OLxP.

Fig. 9. OLTP, OLAP and OLxP performance of tabenchmark.

actional throughput decreases with the analytical request rates increasing under the same transactional request rates. And the average latency of the transactional request rates increases as the analytical request rates increase. The analytical throughput decreases with the incremental transactional request rates when the analytical request rates are less than three tps, and the transactional request rates are less than 7000 tps. The analytical throughput fluctuates wildly when the analytical request rates exceed the processing capacity of the MemSQL. Meanwhile, the long-term running analytical queries increase the waiting time of online transactions.

In the TiDB cluster, under the same transactional request rates, the transactional throughput decreases as the analytical request rates increase when the analytical request rates are no more than 3 tps. Under the same analytical request rates, the analytical throughput decreases as the transactional request rates increase when the analytical request rates are no more than 2 tps, and the transactional request rates are no more than 5000 tps. And the average analytical latency increases as the transactional request rates increase when the transactional request rates are no more than 2500 tps.

C. Tabenchmark evaluation

1) Peak performance: Figure 9(a) shows that the maximum transactional throughput is 124 tps in the MemSQL cluster. The transactional throughput increases as the transactional request rates increase when the transactional request rate is no more than 140 tps. The maximum transactional throughput is 43 tps in the TiDB cluster. The transactional throughput increases as the transactional request rates increase when the transactional request rates are no more than 50 tps. Tabenchmark has the highest percentage of read-only transactions among the three benchmarks. However, there is a slow query in the *DeleteCallForwarding* transaction that took longer than one second, so tabenchmark has the lowest transactional throughput of the three benchmarks. The SQL statement is *"explain SELECT s_id FROM SUBSCRIBER WHERE sub_nbr = ?"*. The s_id and sub_nbr are the composite keys of the SUBSCRIBER table. The full table scan in memory is time-consuming when the slow query is executed in MemSQL. Even worse, when a slow query is executed in the storage engine of the TiDB, the index full scan will perform a random read on the solid-state disk. Therefore, the maximum transactional throughput of MemSQL is higher than the maximum transaction throughput of TiDB. Figure 9(b) shows that the maximum analytical throughput is 0.7 tps in MemSQL cluster. The analytical throughput increases as the analytical request rates increase when the analytical request rates are no more than two tps. And the maximum analytical throughput is 0.23 tps in the TiDB cluster. The analytical throughput increases as the analytical request rates increase when the analytical request rates are no more than two tps.

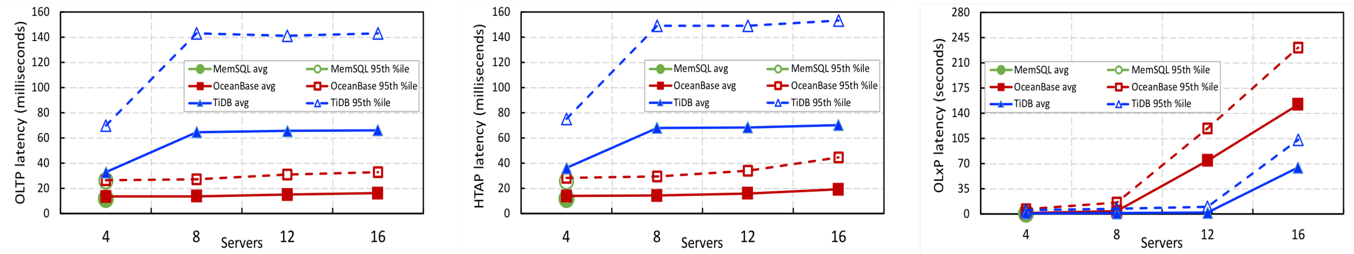
Figure 9(c) shows that the MemSQL cluster is saturated when the hybrid request rate increases to 12 tps. Figure 9(c) shows that the maximum hybrid throughput is around five tps in the TiDB cluster. And the average latency increases with hybrid request rates increases without bound. The gap between 95th percentile latency and average latency is up to 2.2x.

2) Performance interference between OLTP agents and OLAP agents: Figure 9(a) shows that the performance impact of analytical agents on transactional agents. The performance impact of transactional agents on analytical agents is shown in Figure 9(b). In the MemSQL cluster, OLxPBench executes the precisely transactional request rates control when the transactional request rates are no more than 105 tps, and the analytical request rates are no more than three tps. Under the same transactional request rates, the average transactional latency increases more than 34.4 times. And the 95th percentile latency increases by 12.8x. The analytical throughput decreases when the analytical request rates are identical as the transactional request rates rise, which are no more than 50 tps.

The transactional throughput decreases to 49.8% with the analytical agents' inference in the TiDB cluster. It indicates the analytical agents significantly increase the online transaction waiting time. The performance impact of transactional agents on the analytical agents is shown in Figure 9(b). The analytical throughput decreases up to 89% under the transactional agents' inference. The average latency increases 30.8% under the transactional agents' inference. And the 95th percentile latency increases 12.2% under the transactional agents' inference. It indicates that the slow queries in the online transaction block the analytical agents' execution.

D. The main findings of differences between MemSQL and TiDB

First, the enormous transactional performance gap between MemSQL and TiDB results from the different storage mediums for data processing, i.e., memory for MemSQL and



(a) OLTP latency.

(b) OLTP latency with OLAP interference.

(c) OLxP latency.

Fig. 10. OLTP, HTAP and OLxP latency as cluster size increases.

solid-state disk for TiDB. The peak transactional throughput gap between MemSQL and TiDB is 3.0x, 2.6x, and 2.9x using subenchmark, fibenchmark, and tabenchmark. Second, compared to the single storage engine of MemSQL, the separated storage engines (row-based and column-based) of TiDB handle the hybrid workload better, performing real-time queries in-between an online transaction. The peak hybrid workload throughput gap between TiDB and MemSQL is 3.7x and 1.4x using subenchmark and fibenchmark. Third, both MemSQL and TiDB handle the query using the composite keys awkwardly. MemSQL uses time-consuming full table scans in memory, while TiDB uses index full scans that perform a random read on the solid-state disk. The maximum hybrid workload throughput of MemSQL is 2.2x than that of TiDB.

E. Scalability

We choose the mainstream HTAP DBMSs – TiDB [4], MemSQL [14], and OceanBase [15] for scale-out experiments. We test the scaling capability of TiDB and OceanBase by varying the cluster sizes from 4 to 16 nodes [1]. Meanwhile, the data size and target request rates rise in proportion to the increasing cluster size. The following results are the average results of the five runs. TiDB decouples the computational engine layer and storage layer. Due to complex execution plans and compute-intensive queries, we set the ratio of storage instances(SI) to computational instances(CI) at 2:1 in the TiDB cluster. Storage instances are deployed on all servers in the cluster, and computational instances are deployed on half of the servers in the cluster. Oceanbase is shared-nothing architecture, and each OceanBase server (OBServer) is the same. The number of OBServers is the cluster size.

The average latency and 95th percentile latency for workloads in subenchmark are shown in Figure 10. First, OLxPBench clients run on a separate eight vCPU machine and can spawn up to 300 threads to generate target request rates. OLxPBench clients can be deployed on separate client servers, so client servers are not a bottleneck. The more OLxPBench clients are deployed, the more requests are generated. Second, OceanBase and TiDB cannot scale-out well when dealing with the OLTP workloads, OLxP workloads, and the mixtures of OLTP and OLAP workloads. In the OceanBase cluster, the average latency and 95th percentile latency of OLTP workloads increase by 20% and 24% as the cluster size increase from 4 to 16 nodes. In the TiDB cluster, the average latency and 95th percentile latency of OLTP workloads increase more than 1x as the cluster size increase from 4 to 16 nodes. Significantly, the latency of OLxP workloads increases sharply as the cluster size increase from 4 to 16 nodes. It is challenging for the above HTAP DBMSs to deal with the OLxP workloads. Third, compared with OceanBase, TiDB provides better performance isolation as the cluster size increases. Under the same OLAP pressure, the average latency of OLTP workloads increases by 6% and 18% in the TiDB cluster and OceanBase cluster. Meanwhile, TiDB is better than OceanBase at dealing with OLxP workloads. The performance isolation benefits from the decoupled storage layer consisting of a row store (TiKV) and a columnar store (TiFlash).

VII. Conclusions

This paper quantitatively discloses that the previous HTAP benchmarks provide misleading information in evaluating, designing, and implementing HTAP systems. We design and implement an extensible HTAP benchmarking framework named OLxPBench. OLxPBench proposes the abstraction of a hybrid transaction to model the widely-observed behavior pattern – making a quick decision while consulting real-time analysis; a semantically consistent schema to express the relationships between OLTP and OLAP schemas; the combination of domain-specific and general benchmarks to characterize diverse application scenarios with varying resource demands. Extensive experiments demonstrate its merit and pinpoint the bottlenecks of the mainstream distributed HTAP DBMSs.

References

[1] M. Alomari, M. Cahill, A. Fekete, and U. 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.

[2] J. Zhan, "Call for establishing benchmark science and engineering," *BenchCouncil Transactions on Benchmarks, Standards and Evaluations, Volume 1, Issue 1, 100012*, 2021.

[3] J. D. Little and S. C. Graves, "Little's law," in *Building intuition*. Springer, 2008, pp. 81–100.

[4] 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*, vol. 13, no. 12, pp. 3072–3084, 2020.

[5] U. Sirin, S. Dwarkadas, and A. Ailamaki, "Performance characterization of htap workloads," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 1829–1834.

[1] For too high cost for commercial software, we test MemSQL on four servers.

[6] The Apache Hadoop Project. http://hadoop.apache.org/core/.

[7] TPC-C Benchmark. http://www.tpc.org/tpcc/.

[8] Gartner Glossary. https://www.gartner.com/en/information-technology/glossary/real-time-analytics.

[9] TPC-H Benchmark. http://www.tpc.org/tpch/.

[10] TATP Benchmark Description (Version 1.0). http://tatpbenchmark.sourceforge.net.

[11] Real-Time Analytics: The Key to Unlocking Customer Insights Driving the Customer Experience. https://www.gartner.com/doc/3599217/market-guide-htapenabling-inmemory-computing.

[12] Market Guide for HTAP-Enabling In-Memory Computing Technologies. https://www.sas.com/en_za/whitepapers/real-time-analytics-109676.html.

[13] Hybrid Transaction/Analytical Processing Will Foster Opportunities for Dramatic Business Innovation. https://www.gartner.com/en/documents/2657815.

[14] J. Chen, S. Jindel, R. Walzer, R. Sen, N. Jimsheleishvilli, and M. Andrews, "The memsql query optimizer: A modern optimizer for real-time analytics in a distributed database," *Proceedings of the VLDB Endowment*, vol. 9, no. 13, pp. 1401–1412, 2016.

[15] OceanBase. https://github.com/oceanbase/oceanbase

[16] V. Raman, G. Attaluri, R. Barber, N. Chainani, D. Kalmuk, V. KulandaiSamy, J. Leenstra, S. Lightstone, S. Liu, G. M. Lohman *et al.*, "Db2 with blu acceleration: So much more than just a column store," *Proceedings of the VLDB Endowment*, vol. 6, no. 11, pp. 1080–1091, 2013.

[17] D. Butterstein, D. Martin, K. Stolze, F. Beier, J. Zhong, and L. Wang, "Replication at the speed of change: a fast, scalable replication solution for near real-time htap processing," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3245–3257, 2020.

[18] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux, "Oltp-bench: An extensible testbed for benchmarking relational databases," *Proceedings of the VLDB Endowment*, vol. 7, no. 4, pp. 277–288, 2013.

[19] T. Lahiri, M.-A. Neimat, and S. Folkman, "Oracle timesten: An in-memory database for enterprise applications." *IEEE Data Eng. Bull.*, vol. 36, no. 2, pp. 6–13, 2013.

[20] P.-Å. Larson, A. Birka, E. N. Hanson, W. Huang, M. Nowakiewicz, and V. Papadimos, "Real-time analytical processing with sql server," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1740–1751, 2015.

[21] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham *et al.*, "Storm@ twitter," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 147–156.

[22] M. Seltzer, D. Krinsky, K. Smith, and X. Zhang, "The case for application-specific benchmarking," in *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*. IEEE, 1999, pp. 102–107.

[23] T. Lahiri, S. Chavan, M. Colgan, D. Das, A. Ganesh, M. Gleeson, S. Hase, A. Holloway, J. Kamp, T.-H. 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.

[24] D. Makreshanski, J. Giceva, C. Barthels, and 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.

[25] A. Bog, K. Sachs, and H. 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.

[26] F. Özcan, Y. Tian, and P. Tözün, "Hybrid transactional/analytical processing: A survey," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1771–1775.

[27] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Conceptual modeling for etl processes," in *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, 2002, pp. 14–21.

[28] M. Kiran, P. Murphy, I. Monga, J. Dugan, and S. S. Baveja, "Lambda architecture for cost-effective batch and speed big data processing," in *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015, pp. 2785–2792.

[29] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, "Discretized streams: Fault-tolerant streaming computation at scale," in *Proceedings of the twenty-fourth ACM symposium on operating systems principles*, 2013, pp. 423–438.

[30] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*. Ieee, 2010, pp. 1–10.

[31] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi *et al.*, "Spark sql: Relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, 2015, pp. 1383–1394.

[32] A. Kemper and T. Neumann, "Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots," in *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 195–206.

[33] J. Lee, S. Moon, K. H. Kim, D. H. Kim, S. K. Cha, and W.-S. Han, "Parallel replication across formats in sap hana for scaling out mixed oltp/olap workloads," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1598–1609, 2017.

[34] J. Lin, "The lambda and the kappa," *IEEE Internet Computing*, vol. 21, no. 5, pp. 60–66, 2017.

[35] N. Mukherjee, S. Chavan, M. Colgan, M. Gleeson, X. He, A. Holloway, J. Kamp, K. Kulkarni, T. Lahiri, J. 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.

[36] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable real-time data systems*. Manning, 2013.

[37] L. Qu, Q. Wang, T. Chen, K. Li, R. Zhang, X. Zhou, Q. Xu, Z. Yang, C. Yang, W. Qian *et al.*, "Are current benchmarks adequate to evaluate distributed transactional databases?" *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, p. 100031, 2022.

[38] M. J. Cahill, U. Röhm, and A. D. Fekete, "Serializable isolation for snapshot databases," *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 4, pp. 1–42, 2009.

[39] A. Pavlo and M. Aslett, "What's really new with newsql?" *ACM Sigmod Record*, vol. 45, no. 2, pp. 45–55, 2016.

[40] 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.

[41] F. Coelho, J. Paulo, R. Vilaça, J. Pereira, and R. Oliveira, "Htap-bench: Hybrid transactional and analytical processing benchmark," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, 2017, pp. 293–304.

[42] J. Arulraj, A. Pavlo, and P. 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.

[43] M. Athanassoulis, K. S. Bøgh, and S. Idreos, "Optimal column layout for hybrid workloads," *Proceedings of the VLDB Endowment*, vol. 12, no. 13, pp. 2393–2407, 2019.

[44] A. Raza, P. Chrysogelos, A. C. Anadiotis, and A. Ailamaki, "Adaptive htap through elastic resource scheduling," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2043–2054.

[45] J. Yang, I. Rae, J. Xu, J. Shute, Z. Yuan, K. Lau, Q. Zeng, X. Zhao, J. Ma, Z. Chen *et al.*, "F1 lightning: Htap as a service," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3313–3325, 2020.

[46] S. Shen, R. Chen, H. Chen, and B. Zang, "Retrofitting high availability mechanism to tame hybrid transaction/analytical processing," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)(July 2021), USENIX Association*, 2021.

[47] Q. Pi, G. Zhou, Y. Zhang, Z. Wang, L. Ren, Y. Fan, X. Zhu, and K. Gai, "Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2685–2692.

[48] Y. Ma, B. Narayanaswamy, H. Lin, and H. Ding, "Temporal-contextual recommendation in real-time," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2291–2299.