HTAP Databases: What is New and What is Next

Guoliang Li
Department of Computer Science, Tsinghua University
liguoliang@tsinghua.edu.cn

Chao Zhang
Department of Computer Science, Tsinghua University
cycchao@mail.tsinghua.edu.cn

ABSTRACT

Processing the mixed workloads of transactions and analytical queries in a single database system can eliminate the ETL process and enable real-time data analysis on the transaction data. However, there is no free lunch. Such systems must balance the trade-off between workload isolation and data freshness due to interweaving workloads of OLTP and OLAP. Since Gartner coined the term, Hybrid Transactional/Analytical Processing (HTAP), we have witnessed the emergence of various database systems to support HTAP. One common feature is that they leverage the best of row store and column store to achieve high quality of HTAP. As they have disparate storage strategies and processing techniques to satisfy the requirements of various HTAP applications, it is essential to understand, compare, and evaluate their key techniques. In this tutorial, we offer a comprehensive survey of HTAP databases. We introduce a taxonomy of state-of-the-art HTAP databases according to their storage strategies and architectures. We then take a deep dive into their key techniques regarding transaction processing, analytical processing, data synchronization, query optimization, and resource scheduling. We also introduce existing HTAP benchmarks. Finally, we discuss the research challenges and open problems for HTAP.

CCS CONCEPTS

• **Information systems** → **Database transaction processing**; **Database query processing**.

KEYWORDS

HTAP Databases; Transaction Processing; Query Processing

ACM Reference Format:
Guoliang Li and Chao Zhang. 2022. HTAP Databases: What is New and What is Next. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, Philadelphia, PA, USA, 6 pages. https://doi.org/10.1145/3514221.3522565

1 INTRODUCTION

Background. All organizations are processing more data than ever at their disposal, and data keeps coming with high velocity, volume and variety [26, 30, 53, 55]. For businesses with data-intensive applications, it is beneficial to have a single HTAP system that not only can efficiently handle on-line transactional processing (OLTP), but also can perform on-line analytical processing (OLAP) for prompt decision-making. For instance, when equipped with an HTAP system, entrepreneurs in retail applications can analyze the latest transaction data in real time and identify the sales trend, then take timely actions, e.g., roll out advertising campaigns for promising products [35]. In finance applications, vendors can leverage an HTAP system to process the customer transactions efficiently while detecting the fraudulent transactions simultaneously [16, 36, 47].

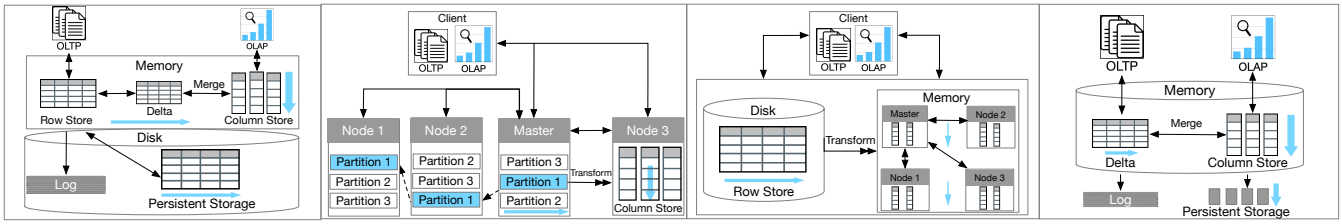
HTAP Definition. Hybrid Transactional/Analytical Processing (HTAP) is an application architecture proposed by a Gartner report [35] at 2014, which utilizes in-memory computing technologies to enable concurrent analytical and transaction processing on the same in-memory data store. Such an architecture should eliminate the need of Extract-Transform-Load (ETL) process, thereby accelerating data analytics and bringing dramatic business innovation. In 2018, Gartner extended the HTAP concept to "In-Process HTAP" [15], an application architecture that supports weaving analytical and transaction processing techniques together as needed to accomplish the business task. Such a new definition indicates HTAP is no longer limited to in-memory computing techniques.

Motivation. Over the last few years, numerous database systems [18–22, 29, 31, 42, 44] have been developed to enable HTAP. One common feature is that they utilize the best of row store and column store to achieve high quality of HTAP. Nevertheless, they have disparate storage strategies and processing techniques albeit the dual-store feature. This main reason for such diversity is that different classes of HTAP systems target at different applications. For instance, it depends on whether OLTP or OLAP is the first citizen of the applications, or both are important. It also depends on the requirements of availability, scalability, system performance, and data freshness [9] specified in the service level agreements (SLAs) [17]. Consequently, HTAP systems must balance the trade-off between workload isolation and data freshness due to interweaving workloads of OLTP and OLAP. To better harness these HTAP forces for various applications, it is of paramount importance to study, understand, and compare their key techniques. In this tutorial, we study HTAP databases that utilize row store and column store together to efficiently handle the mixed workloads of OLTP and OLAP in a single database system.

Tutorial Overview. We will provide a comprehensive tutorial on HTAP databases. The intended length of the tutorial is 3 hours. The tutorial consists of four sections as follows.
(1) HTAP Databases (30 min). This section starts with an introduction to the background of HTAP databases. It provides a classification according to their storage architectures, then introduces the main approaches in each category. As shown in Figure 1, it classifies HTAP databases into four categories: (a) Primary Row store + In-Memory Column store; (b) Distributed Row Store + Column Store Replica; (c) Disk Row Store + Distributed Column Store; and (d) Primary Column Store + Delta Row Store. Then, it presents the main HTAP techniques and representatives for each architecture.

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA.

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3522565



(a) Primary Row Store+In-Memory Column Store (b) Distributed Row Store + Column Store Replica (c) Disk Row Store + Distributed Column Store (d) Primary Column Store + Delta Row Store

Figure 1: Storage Architectures of State-Of-The-Art HTAP Databases

Table 1: A Classification of State-Of-The-Art HTAP Databases based on the Storage Architecture

Category	HTAP databases	TP Throughput	AP Throughput	TP Scalability	AP Scalability	Isolation	Freshness
Primary Row Store + In-Memory Column Store	Oracle Dual-Format[19], SQL Server[20], DB2 BLU[39]	High	High	Medium	Low	Low	High
Distributed Row Store + Column Store Replica	TiDB[18], SingleStore[44]	Medium	Medium	High	High	High	Low
Disk Row Store + Distributed Column Store	MySQL Heatwave[31]	Medium	Medium	Medium	High	High	Medium
Primary Column Store + Delta Row Store	SAP HANA[43]	Medium	High	Low	Medium	Low	High

Particularly, it summarizes the pros and cons of different HTAP solutions regarding performance, scalability, workload isolation, and data freshness (see Table 1).

(2) HTAP Techniques (40 min). This section takes a deep dive into the key techniques of HTAP databases, paying particular attentions to their techniques concerning transaction processing, analytical processing, data synchronization, query optimization, and resource scheduling. The detailed key techniques in each module are shown in Table 2. Overall, it focuses on five task types for HTAP as follows.

– *Transaction processing (TP) techniques.* This part will introduce two types of TP techniques, including (i) MVCC + logging [19, 20, 31, 39, 43] that relies on multi-version concurrency control (MVCC) protocols and logging techniques for transaction processing; and (ii) 2PC+Raft+logging [18] that processes the transactions in a distributed architecture based on a two-phase commit (2PC) protocol, a Raft-based consensus algorithm, and logging techniques.

– *Analytical processing (AP) techniques.* This part will introduce three kinds of AP techniques. The first type is (i) in-memory delta and column scan [19, 20, 31, 39, 43] that responds to an analytical query by performing a scan on the in-memory columnar data and visible delta tuples yet being merged simultaneously. The second type is (ii) disk-based delta and column scan [18] that scans the log-based delta files and the column store together for an incoming query. The third type is (iii) column scan [44] that performs the query purely in the column store.

– *Data synchronization (DS) techniques.* This part will introduce three types of DS techniques for synchronizing data between OLTP and OLAP, including (i) in-memory delta merge [19, 20, 31, 39, 43] that merges the newly-inserted in-memory delta data to the main column store; and (ii) disk-based delta merge [18] that periodically merges the disk-based delta files to the main column store; and (iii) rebuild from primary row store [19, 39] that rebuilds the in-memory column store from the primary row store.

– *Query optimization techniques.* This part will introduce three aspects of query optimization techniques, including (i) column selection for HTAP [19, 31] that automatically selects the columns from the primary store into main memory based on the history workload; (ii) hybrid row/column scan [18, 20] that relies on cost-based functions to determines whether to perform a query over the row store or over the column store; and (iii) CPU/GPU Acceleration for HTAP [5, 22] that leverages heterogeneous hardware, i.e., CPU/GPU architecture to accelerate HTAP workloads, respectively.

– *Resource scheduling techniques.* This part will introduce the resource scheduling techniques that aim to improve the resource utilization by dynamically allocating resources, e.g., CPU and memory, for HTAP. It mainly introduces two types of techniques. The first one is the workload-driven scheduling [43, 45] that adaptively adjusts the resources of OLTP and OLAP workloads based on the execution status of workload. The second one is the freshness-driven scheduling [40] that controls the execution modes of HTAP workloads based on the freshness metric.

(3) HTAP Benchmarks (10 mins). This section introduces the existing benchmarks and evaluation practices on HTAP databases. It will introduce several end-to-end HTAP benchmarks including TPC-C [48], TPC-H [49], HTAPbench [10], and CH-benchmark [11]. Specifically, it will walk through the key aspects of the benchmarks, including data generation, execution rule, and performance metrics. In addition, it will introduce two HTAP micro-benchmarks: ADAPT [6] and HAP [7] benchmarks. After that, it summarizes the key insights from existing evaluation practices [13, 38, 40, 42, 45].

(4) Challenges and Open Problems (10 mins). The final section concludes the tutorial and discusses the research challenges and open problems for HTAP techniques. It summarizes the tutorial topics, then presents several challenges and open problems. Firstly, it presents the limitations of existing methods on column selection for HTAP workloads, then discusses the possibility of learning-based methods on this task. Secondly, it discusses the challenges for HTAP query optimization and calls for a learned query optimizer for HTAP. Thirdly, it discusses the limitation of current approaches on HTAP resource scheduling, then calls for new adaptive methods. Finally, it discusses the limitation of existing benchmarks and envisions a new HTAP benchmark suite.

Difference with Existing Tutorial. There has been an 1.5-hour tutorial [34] on Hybrid Transactional/Analytical Processing (HTAP) in SIGMOD 2017. It gave a general classification to HTAP systems including NoSQL databases [1, 2], Hadoop-based data warehouses [3, 4], and loosely-coupled Spark-based [52] systems. Different from the previous tutorial, we provide a new taxonomy of HTAP databases that should support ACID-compliant transaction processing and real-time analytical processing simultaneously. In addition, we introduce HTAP databases based on the latest developments, i.e., the database systems and techniques that emerged since 2017. We also go deeper into the fundamental techniques of HTAP databases and summarize the pros and cons of different approaches. Last but not least, we introduce the benchmarks and evaluation practices for HTAP, which have not yet been presented.

Target Audience. This tutorial is intended for a wide scope of SIGMOD attendees, including entrepreneurs, researchers, developers, practitioners, and students. For entrepreneurs, this tutorial helps them gain a comprehensive picture and understanding of the state-of-the-art HTAP techniques that may be suitable for their business cases. For researchers, they can gain insights from the pros and cons of existing techniques, find new topics and research problems, and contribute their expertise to HTAP databases. For developers and practitioners, this tutorial could deepen their understanding of key techniques of HTAP databases and benchmarks, which helps them to properly choose or improve the HTAP systems for their applications. For students, this tutorial provides them with crucial techniques on HTAP.

2 TUTORIAL OUTLINE

We start with an introduction to the background of HTAP, and then summarize a taxonomy of HTAP databases. We introduce their main approaches, discuss the pros and cons, then go deeper into the key techniques with illustrated examples. Furthermore, we introduce HTAP benchmarks and evaluation practices. Finally, we discuss research challenges and open problems.

2.1 HTAP Databases

In this tutorial, we study HTAP databases [18–20, 29, 31, 32, 37, 39, 43, 44] that leverage the row store and column store together to enable HTAP in a single database system.

(a) Primary Row Store+In-Memory Column Store. This category of HTAP databases [19, 20, 29, 37, 39] leverage primary row store as the basis for OLTP workloads and processes OLAP workloads with an in-memory column store. All the data is persisted to the primary row store. The row store is also memory-optimized such that data updates are efficiently handled. Updates are also appended to the delta store which will be merged to the column store. For instance, Oracle in-memory dual-format database [19] combines the row-based buffer and column-based in-memory compression unit (IMCU) to handle OLTP and OLAP workloads together. IMCU is populated from the buffer and changes are cached in the snapshot metadata unit (SMU). Another example is SQL Server [20] that developed the columnstore index (CSI) over in-memory tables in the Hekaton [12] row engine to enable real-time analytical processing. HTAP databases in this type have high throughput as all the workloads are processed in memory.

(b) Distributed Row Store+Column Store Replica. This category relies on distributed architecture to support HTAP. The master node asynchronously replicates the logs to the slave nodes when handling the transaction requests. The primary storage is row store, some slave nodes will be chosen as column store servers for query acceleration. Transactions are handled in a distributed way for high scalability; complex queries are performed in the server nodes with a column store. A representative is TiDB [18], which is a Raft-based distributed HTAP database that asynchronously replicates Raft logs from the leader node to follower nodes storing the data in the row-based replicas. The logs are also sent to learner nodes that store the data in columnar format. As a result, the workload isolation level is high as transactions are processed on nodes with a row store, and analytical queries are executed on nodes with a column store. However, the data freshness is low since newly-updated data may have not been merged to the column store.

(c) Disk Row Store+Distributed Column Store. This kind of databases utilizes a disk-based RDBMS with a distributed in-memory column-store (IMCS) to support HTAP. The RDBMS preserves the full capacity for OLTP workloads and an IMCS cluster is deeply integrated to accelerate query processing. The columnar data is extracted from the row store, hot data resides in IMCS and cold data will be evicted to disk. For instance, MySQL Heatwave [31] combines a MySQL database with a distributed IMCS cluster, called Heatwave, to enable real-time analytics. Transactions are fully executed in the MySQL database. Columns that are frequently accessed will be loaded into the Heatwave. When a complex query comes in, it can be pushed down to the IMCS engine for query acceleration.

(d) Primary Column Store+Delta Row Store. This category of databases utilizes primary column store as the basis for OLAP, and handles OLTP with a delta row store. The in-memory delta-main HTAP databases store the whole data in the main column store. Data updates are appended to the row-based delta store. The OLAP performance is high as the column store is highly read-optimized. However, since there is only a delta row store for OLTP workloads, the OLTP scalability is low. A representative is SAP HANA [14, 43]. It divides the in-memory data store into three layers: L1-delta, L2-delta, and Main. The L1-delta keeps data updates in a row-wise format. When the threshold is reached, the data in L1-delta is appended to L2-delta. The L2-delta transforms the data into columnar data, then merges the data into the main column store. Finally, the columnar data is persisted to the disk storage.

2.2 HTAP Techniques

As shown in Table 2, we summarize five task types of HTAP techniques, including (1) transaction processing; (2) analytical processing; (3) data synchronization; (4) query optimization; and (5) resource scheduling. These key techniques are adopted by the state-of-the-art HTAP databases. Nevertheless, they have pros and cons concerning various metrics, e.g., efficiency, scalability, and freshness. At the end of the section, we review other related HTAP techniques that complement the key techniques.

(1) Transaction Processing (TP) Techniques. OLTP workloads in HTAP databases are handled over the row store, but different architectures result in disparate TP techniques. It mainly consists of two types. The first type is (i) MVCC+logging [19, 20, 31, 39, 43]

Table 2: An Overview of HTAP Techniques

Task Type	Key Techniques	HTAP Databases/Prototypes	Pros	Cons
Transaction Processing	MVCC+ Logging	Oracle Dual-Format[19] SQL Server[20], DB2 BLU[39] MySQL Heatwave[31], SAP HANA[43]	High Efficiency	Low Scalability
	2PC+Raft+Logging	TiDB[18]	High Scalability	Low Efficiency
Analytical Processing	In-Memory Delta and Column Scan	Oracle Dual-Format[19] SQL Server[20], DB2 BLU[39] MySQL Heatwave[31], SAP HANA[43]	High Freshness	Large Memory Size
	Log-based Delta and Column Scan	TiDB[18]	High Scalability	Low Freshness
	Column Scan	SingleStore[44]	High Efficiency	Low Freshness
Data Synchronization	In-Memory Delta Merge	Oracle Dual-Format[19] SQL Server[20], DB2 BLU[39] MySQL Heatwave[31], SAP HANA[43]	High Efficiency	Low Scalability
	Log-based Delta Merge	TiDB[18]	High Scalability	High Merge Cost
	Rebuild from Primary Row Store	SingleStore[44], Oracle Dual-Format[19]	Small Memory Size	High Load Cost
Query Optimization	In-Memory Column Selection	Oracle 21c [33], MySQL Heatwave[31]	High Memory Utility	Low AP Throughput
	Hybrid Row/Column Scan	TiDB[18], SQL Server[13, 20]	High AP Throughput	Large Search Space
	CPU/GPU Acceleration for HTAP	RateupDB [22], Caldera [5]	High AP Throughput	Low TP Throughput
Resource Scheduling	Freshness-driven scheduling for HTAP	RDE [40]	High Freshness	Low Throughput
	Workload-driven scheduling for HTAP	SAP HANA[43], Siper[45]	High Throughput	Low Freshness

that relies on MVCC protocols and logging techniques to process the transactions. Specifically, each insert is first written to the log and the row store, then is appended to the in-memory delta store. An update creates a new version of a row with a new lifetime of a begin timestamp and an end timestamp, the older version is marked as a delete row in a delete bitmap. As a result, the transaction processing is efficient as the DML operations are performed in memory. Note that some approaches may write data to either the row store [44] or the delta row store [43], and they may only write log when the transaction is committing [12, 20]. The second type is (ii) 2PC+Raft+logging [18] that relies on distributed architecture (b) introduced in Section 2.1. It provides high scalability with distributed transaction processing. The ACID transactions are processed on the distributed nodes with a 2-phase commit (2PC) protocol, a Raft-based consensus algorithm, and the write-ahead log (WAL) technique. Particularly, the leader node receives the request from the SQL engine, then appends logs locally and sends logs to follower nodes asynchronously. If the majority of nodes, i.e., the quorum, successfully append the logs, the leader commits the request and applies it locally. Compared to the first type, the second type has low efficiency due to distributed transaction processing.

(2) Analytical Processing (AP) Techniques. For HTAP databases, OLAP workloads are performed using column-oriented techniques such as aggregations over compressed data and single-instruction multiple-data (SIMD) instructions [39, 43]. Particularly, they are divided into three types. The first type is (i) in-memory delta and column scan [19, 20, 31, 39, 43]. This line of work scans the in-memory delta and columnar data together as the delta store may include the updated records that have not been merged to the column store. Since it has scanned the recently visible delta tuples in memory, the data freshness is high for OLAP. The second type is (ii) log-based delta and column scan [18] that scans the log-based delta data and the columnar data together for incoming queries. Similar to the first type, the second type scans the latest delta with the column store for handling OLAP. However, such a process is more expensive due to reading the delta files that may have not been merged. Consequently, the data freshness is low due to the high latency of shipping and merging the delta files. The third type is (iii) column scan [44], which only scans the columnar data for high efficiency as there is no overhead of reading any delta data. However, this technique leads to low freshness when the data is frequently updated in the row store.

(3) Data Synchronization (DS) Techniques. Since reading the delta data in query time is expensive, it is necessary to periodically merge the delta data to the main column store. There are three kinds of DS techniques for various HTAP databases. Namely, (i) in-memory delta merge [19, 20, 31, 39, 43]; (ii) disk-based delta merge [18]; and (iii) rebuild from primary row store [19, 39]. The first category periodically merges the newly-inserted in-memory delta data to the main column store. Several techniques are introduced to optimize the merge process, including the two-phase transaction-based data migration [20], the dictionary-encoded sorting merge [43], and threshold-based change propagation [19, 31, 39]. The second category [18] merges the disk-based delta files to the main column store. To speed up the merging process, the delta data can be indexed by a B+-tree, thus the delta items can be efficiently located with key lookups [18]. The third category rebuilds the in-memory column store from the primary row store [19, 44]. This is typical for the case that the delta updates exceed a certain threshold, thus it is more efficient to rebuild the column store than merging these updates with a large memory footprint [19].

(4) Query Optimization Techniques. We introduce three aspects of query optimization techniques, including (i) column selection for HTAP [31, 33]; (ii) hybrid row/column scan [18, 20]; and (iii) CPU/GPU acceleration for HTAP [5, 22]. The first type [31, 33] relies on history workload and statistics to select frequently-accessed columns extracted from the primary store into memory. Thus, a query can be pushed down to the in-memory column store for acceleration. The downside is that the columns for a new query may have not been selected, leading to row-based query processing. Existing approaches rely on history workload's access pattern [31, 33] to load the hot data and evict the cold data. The second type [18, 20] utilizes hybrid row/column scan to accelerate a query.

With such techniques, a complex query can be decomposed to perform either over the row store or over the column store, then the results are combined. This is typical for an SPJ query that can be executed with a row-based index scan and a complete column-based scan. We introduce the cost-based approach [18, 20] to select the hybrid row/column access path. The third type of techniques [5, 22] leverages heterogeneous CPU/GPU architecture to accelerate HTAP workloads. These techniques utilize the task-parallel nature of CPUs and the data-parallel nature of GPUs for handling OLTP and OLAP, respectively. Nevertheless, such techniques favor high OLAP throughput while having low OLTP throughput.

(5) Resource Scheduling Techniques. For HTAP databases, resource scheduling refers to resource allocation for OLTP and OLAP workloads. Existing techniques [40, 43, 45] dynamically control the execution mode of OLTP and OLAP workloads for better resource utilization. There are two types of scheduling techniques, the workload-driven approaches [43, 45] and the freshness-driven approach [40]. The former one adjusts the parallelism threads of OLTP and OLAP tasks based on the performance of executed workloads. For example, when CPU resource is saturated by OLAP threads, the task scheduler can decrease the parallelism of OLAP while enlarging the OLTP threads. The latter one [40] switches the execution modes on resource allocation and data exchange for OLTP and OLAP. For instance, the scheduler controls the execution of OLTP and OLAP in isolation for high throughput, then periodically synchronizes the data. Once the data freshness becomes low, it switches to an execution mode with shared CPU, memory and data.

Other HTAP-related techniques. We also review (i) new HTAP indexing techniques [41, 46] and (ii) scale-out techniques[17, 19].

2.3 HTAP Benchmarks

We present existing benchmarks [11, 48, 49] and evaluation practices [13, 38, 40, 42, 45] on HTAP databases.

(1) HTAP Benchmarks. A widely-used end-to-end HTAP benchmark is CH-benchmark [11] that combines two TPC benchmarks, i.e., TPC-C [48] for transactional workloads, TPC-H [49] for analytical workloads. Another end-to-end benchmark, called HTAPBench [10], also combines TPC-C and TPC-H, but proposes a different metric. We will compare HTAPBench with CH-benchmark concerning data generation, execution rule, and performance metrics. For data generation, we study how they scale the original data generator. For the execution rule, we present how they control the concurrent execution of OLTP and OLAP workloads with benchmark parameters. For performance metrics, we introduce how they combine the metrics of transactions per minute (tpmC) and completed queries per hour (QphH). In addition to the end-to-end benchmarks, we will introduce two synthetic micro-benchmarks [6, 7].

(2) HTAP Database Evaluation. We summarize the gained insights from existing evaluation practices on HTAP databases [13, 38, 40, 42, 45]. Particularly, we study the trade-offs that HTAP systems made for handling the OLTP and OLAP workloads. We will present quantified numbers to shed some light on how different HTAP databases perform in various situations. For example, to strike the trade-off between workload isolation and data freshness, we compare what percentage of performance degradation the systems should pay in order to maintain the data freshness.

2.4 Challenges and Open Problems

Automatic Column Selection for HTAP Workload. Given an HTAP workload, selecting which columns into the in-memory column store from the row store is an important task. However, existing methods rely heavily on the historical statistics to select the columns into memory, e.g., Oracle 21c's Heatmap [33]. Such methods make the recommendation by running all the queries, thus are expensive and inflexible. Recently, learning-based methods [24] have been widely used in the database field, including knob tuning [25, 54], join ordering [50], and view selection[51]. Therefore, it calls for new automatic methods to efficiently and effectively select the columns for HTAP workloads. The main challenge is to design a lightweight learned method that can capture the access patterns of workloads without executing the entire workload in the database. In addition, it is challenging to take into account the data encoding together, which leads to a larger search space.

Learned HTAP Query Optimizer. Existing methods [13, 18] optimize the query by leveraging cost functions to select the access path of row store and column store in an HTAP database. However, they make uniform and independent assumptions to estimate the row/column size, then use such estimates to measure the scan cost for row store and column store. Such methods are problematic for correlated and skewed data due to inaccurate cost estimates. Recently, learned query optimizers [27, 28] have shown practical gains by learning a mapping between an incoming query and the execution strategy from the existing optimizer. Therefore, it is also promising to develop a learned query optimizer for HTAP databases. The main challenge is to consider both the row-based and column-based operators in the learning phase as the learning space is large.

Adaptive HTAP Resource Scheduling. HTAP resource scheduling helps databases to balance the trade-off between workload isolation and data freshness. This is achieved by adjusting the execution modes of OLAP and OLTP. Isolated execution of OLAP and OLTP workloads favors high throughput but has a low data freshness. Shared execution of mixed workloads favors a high data freshness but has strong workload interference. Exiting freshness-driven scheduling [40] relies on a rule-based approach to control the execution mode but neglects the workload pattern. The workload-driven approach [43, 45] adjusts the threads of OLTP and OLAP but does not consider the freshness. Thus, it is important to consider both workload and freshness when scheduling the resources. To this end, it is preferable to develop a lightweight adaptive scheduling method that not only captures the workload pattern for better performance, but also satisfies the requirement of data freshness.

HTAP Benchmark Suite. First, it has been pointed out [8, 23] that TPC-H has a uniform distribution with little correlation across columns, posing a little challenge on testing OLAP. Thus, HTAP benchmarks with TPC-H should incorporate the join-crossing correlation with skew into the benchmark. Second, Gartner has defined [34, 35] HTAP transaction could contain analytical operations. However, this feature is still not introduced in any HTAP benchmark. Therefore, it calls for a new HTAP benchmark with analytical operations, e.g., insert analytical operations to TPC-C. Third, there is a dearth of specific micro-benchmarks for HTAP tasks, e.g., data synchronization, query optimization, and resource scheduling. All

in all, it calls for a new testbed that can extend various components of existing benchmarks for a holistic evaluation of HTAP databases.

3 BIOGRAPHY

Guoliang Li is a full professor at the Department of Computer Science, Tsinghua University. His research interests include database systems, large-scale data cleaning and integration. He got VLDB 2017 early research contribution award and TCDE 2014 early career award. He will present HTAP databases and open problems.

Chao Zhang is a postdoctoral researcher at Tsinghua University. He received his PhD degree from University of Helsinki, Finland. His research interests focus on database management systems. He will present the HTAP techniques and benchmarks.

4 ACKNOWLEDGEMENT

This paper was supported by NSF of China (61925205), Huawei, TAL education, and Beijing National Research Center for Information Science and Technology (BNRist).

REFERENCES

[1] Apache Cassandra. Apache Cassandra– An Open Source NoSQL Distributed Database, 2021.
[2] Apache HBase. A Distributed, Scalable, Big Data Store., 2021.
[3] Apache Hive. A Data Warehouse using SQL., 2021.
[4] Apache Impala. An Open Source, Native Analytic Database for Hadoop., 2021.
[5] R. Appuswamy, M. Karpathiotakis, D. Porobic, and A. Ailamaki. The Case For Heterogeneous HTAP. In *CIDR*, 2017.
[6] J. Arulraj, A. Pavlo, and P. Menon. Bridging the Archipelago between Row-stores and Column-stores for Hybrid Workloads. In *SIGMOD*, pages 583–598, 2016.
[7] M. Athanassoulis, K. S. Bøgh, and S. Idreos. Optimal Column Layout for Hybrid Workloads. *Proceedings of the VLDB Endowment*, 12(13):2393–2407, 2019.
[8] P. Boncz, A.-C. Anatiotis, and S. Kläbe. JCC-H: Adding Join Crossing Correlations with Skew to TPC-H. In *Technology Conference on Performance Evaluation and Benchmarking*, pages 103–119. Springer, 2017.
[9] M. Bouzeghoub. A Framework for Analysis of Data Freshness. In *International workshop on Information quality in information systems*, pages 59–67, 2004.
[10] F. Coelho, J. Paulo, R. Vilaça, J. Pereira, and R. Oliveira. HTAPBench: Hybrid Transactional and Analytical Processing Benchmark. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, pages 293–304, 2017.
[11] R. Cole, F. Funke, L. Giakoumakis, W. Guy, A. Kemper, S. Krompass, H. Kuno, R. Nambiar, T. Neumann, M. Poess, et al. The Mixed Workload CH-benCHmark. In *Proceedings of the Fourth International Workshop on Testing Database Systems*, pages 1–6, 2011.
[12] C. Diaconu, C. Freedman, E. Ismert, P.-A. Larson, P. Mittal, R. Stonecipher, N. Verma, and M. Zwilling. Hekaton: SQL Server's Memory-Optimized OLTP Engine. In *SIGMOD*, pages 1243–1254, 2013.
[13] A. Dziedzic, J. Wang, S. Das, B. Ding, V. R. Narasayya, and M. Syamala. Columnstore and B+ tree-Are Hybrid Physical Designs Important? In *SIGMOD*, pages 177–190, 2018.
[14] F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, and J. Dees. The SAP HANA Database–An Architecture Overview. *IEEE Data Eng. Bull.*, 35(1):28–33, 2012.
[15] D. Feinberg. Setting the Record Straight – HTAP OPDBMS, 2018.
[16] B. Gallet and M. Gowanlock. Heterogeneous CPU-GPU epsilon grid joins: Static and dynamic work partitioning strategies. *Data Sci. Eng.*, 6(1):39–62, 2021.
[17] A. K. Goel, J. Pound, N. Auch, P. Bumbulis, S. MacLean, F. Färber, F. Gropengiesser, C. Mathis, T. Bodner, and W. Lehner. Towards Scalable Real-Time Analytics: An Architecture for Scale-Out of OLxP Workloads. *Proceedings of the VLDB Endowment*, 8(12):1716–1727, 2015.
[18] D. Huang, Q. Liu, Q. Cui, Z. Fang, X. Ma, F. Xu, L. Shen, L. Tang, Y. Zhou, M. Huang, et al. TiDB: A Raft-based HTAP Database. *Proceedings of the VLDB Endowment*, 13(12):3072–3084, 2020.
[19] 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*, pages 1253–1258. IEEE, 2015.
[20] P.-Å. Larson, A. Birka, E. N. Hanson, W. Huang, M. Nowakiewicz, and V. Papadimos. Real-Time Analytical Processing with SQL Server. *VLDB*, 8(12):1740–1751, 2015.
[21] 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. *VLDB*, 10(12):1598–1609, 2017.
[22] R. Lee, M. Zhou, C. Li, S. Hu, J. Teng, D. Li, and X. Zhang. The Art of Balance: A RateupDB Experience of Building a CPU/GPU Hybrid Database Product. *VLDB*, 14(12):2999–3013, 2021.
[23] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How Good Are Query Optimizers, Really? *VLDB*, 9(3):204–215, 2015.
[24] G. Li, X. Zhou, and L. Cao. AI Meets Database: AI4DB and DB4AI. In *SIGMOD*, pages 2859–2866, 2021.
[25] G. Li, X. Zhou, S. Li, and B. Gao. Qtune: A Query-Aware Database Tuning System with Deep Reinforcement Learning. *VLDB*, 12(12):2118–2130, 2019.
[26] G. Li, X. Zhou, J. Sun, X. Yu, Y. Han, L. Jin, W. Li, T. Wang, and S. Li. opengauss: An autonomous database system. *Proc. VLDB Endow.*, 14(12):3028–3041, 2021.
[27] R. Marcus, P. Negi, H. Mao, N. Tatbul, M. Alizadeh, and T. Kraska. Bao: Making Learned Query Optimization Practical. In *SIGMOD*, pages 1275–1288, 2021.
[28] R. C. Marcus, P. Negi, H. Mao, C. Zhang, M. Alizadeh, T. Kraska, O. Papaemmanouil, and N. Tatbul. Neo: A Learned Query Optimizer. *VLDB*, 12(11):1705–1718, 2019.
[29] MariaDB. Deploy an HTAP Server with MariaDB ColumnStore 5.5 and Community Server 10.6, 2021.
[30] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton. Big Data: The Management Revolution. *Harvard business review*, 90(10):60–68, 2012.
[31] MySQL Heatwave. Real-time Analytics for MySQL Database Service, 2021.
[32] T. Neumann, T. Mühlbauer, and A. Kemper. Fast Serializable Multi-Version Concurrency Control for Main-Memory Database Systems. In *SIGMOD*, pages 677–689, 2015.
[33] Oracle 21c. Automating Management of In-Memory Objects., 2021.
[34] F. Özcan, Y. Tian, and P. Tözün. Hybrid Transactional/Analytical Processing: A Survey. In *SIGMOD*, pages 1771–1775, 2017.
[35] M. Pezzini, D. Feinberg, N. Rayner, and R. Edjlali. Hybrid Transaction/Analytical Processing Will Foster Opportunities For Dramatic Business Innovation. *Gartner (2014, January 28)*, pages 4–20, 2014.
[36] M. Pezzini, D. Feinberg, N. Rayner, and R. Edjlali. Real-time Insights and Decision Making using Hybrid Streaming, In-Memory Computing Analytics and Transaction Processing. 2016.
[37] PolarDB. PolarDB HTAP Real-Time Data Analysis Technology Decryption, 2021.
[38] I. Psaroudakis, F. Wolf, N. May, T. Neumann, A. Böhm, A. Ailamaki, and K.-U. Sattler. Scaling Up Mixed Workloads: A Battle of Data Freshness, Flexibility, and Scheduling. In *Technology Conference on Performance Evaluation and Benchmarking*, pages 97–112. Springer, 2014.
[39] 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. *VLDB*, 6(11):1080–1091, 2013.
[40] A. Raza, P. Chrysogelos, A. C. Anadiotis, and A. Ailamaki. Adaptive HTAP Through Elastic Resource Scheduling. In *SIGMOD*, pages 2043–2054, 2020.
[41] C. Riegger, T. Vinçon, R. Gottstein, and I. Petrov. MV-PBT: Multi-Version Indexing for Large Datasets and HTAP Workloads. In *EDBT*, pages 217–228, 2020.
[42] S. Shen, R. Chen, H. Chen, and B. Zang. Retrofitting High Availability Mechanism to Tame Hybrid Transaction/Analytical Processing. In *OSDI*, pages 219–238, 2021.
[43] V. Sikka, F. Färber, W. Lehner, S. K. Cha, T. Peh, and C. Bornhövd. Efficient Transaction Processing in SAP HANA Database: The End of A Column Store Myth. In *SIGMOD*, pages 731–742, 2012.
[44] SingleStore. The Single Database for All Data-Intensive Applications, 2021.
[45] U. Sirin, S. Dwarkadas, and A. Ailamaki. Performance Characterization of HTAP Workloads. In *ICDE*, pages 1829–1834. IEEE, 2021.
[46] Y. Sun, G. E. Blelloch, W. S. Lim, and A. Pavlo. On Supporting Efficient Snapshot Isolation for Hybrid Workloads with Multi-Versioned Indexes. *VLDB*, 13(2), 2019.
[47] B. Tran, B. Schaffner, J. M. Myre, J. Sawin, and D. Chiu. Exploring means to enhance the efficiency of GPU bitmap index query processing. *Data Sci. Eng.*, 6(2):209–228, 2021.
[48] Transaction Processing Performance Council. TPC-C, 2021.
[49] Transaction Processing Performance Council. TPC-H, 2021.
[50] X. Yu, G. Li, C. Chai, and N. Tang. Reinforcement Learning with Tree-LSTM for Join Order Selection. In *ICDE*, pages 1297–1308. IEEE, 2020.
[51] H. Yuan, G. Li, L. Feng, J. Sun, and Y. Han. Automatic View Generation with Deep Learning and Reinforcement Learning. In *ICDE*, pages 1501–1512, 2020.
[52] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.
[53] C. Zhang, J. Lu, P. Xu, and Y. Chen. UniBench: A Benchmark for Multi-Model Database Management Systems. In *TPCTC*, volume 11135 of *Lecture Notes in Computer Science*, pages 7–23. Springer, 2018.
[54] J. Zhang, Y. Liu, K. Zhou, G. Li, Z. Xiao, B. Cheng, J. Xing, Y. Wang, T. Cheng, L. Liu, et al. An End-To-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning. In *SIGMOD*, pages 415–432, 2019.
[55] X. Zhou, C. Chai, G. Li, and J. Sun. Database meets artificial intelligence: A survey. *IEEE Trans. Knowl. Data Eng.*, 34(3):1096–1116, 2022.