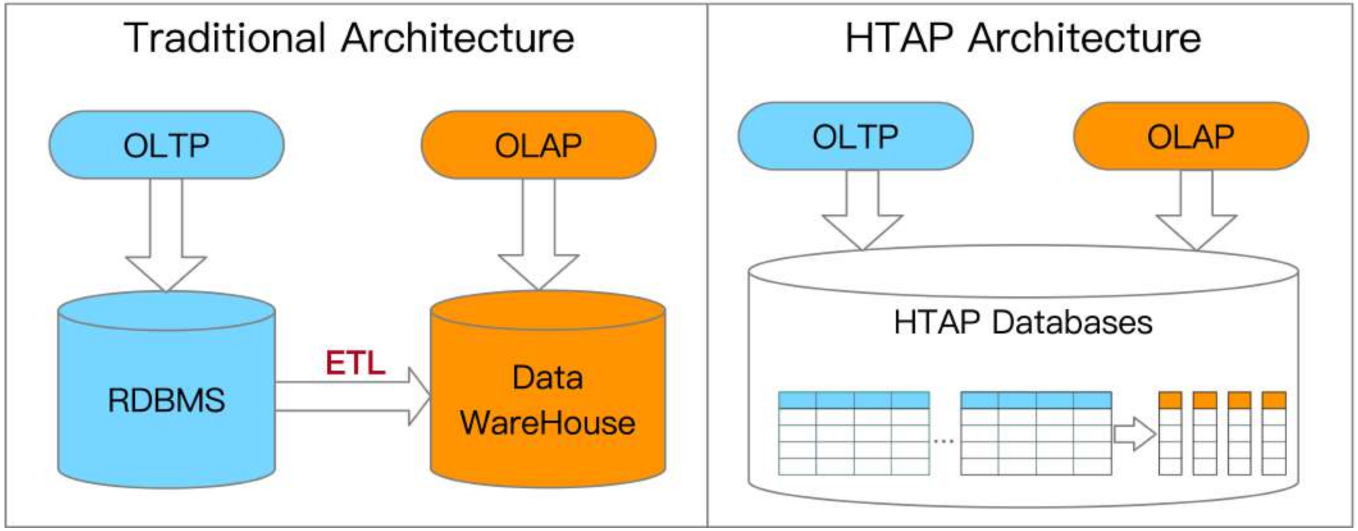
HTAP Databases: A Tutorial

Guoliang Li
Tsinghua University

Chao Zhang
Tsinghua University

Motivation

- **Hybrid Transactional and Analytical Processing, HTAP**
  - **Gartner's definition in 2014:** utilizes in-memory computing technologies to enable concurrent analytical and transaction processing on the same in-memory data store
  - **Gartner's new definition in 2018:** supports weaving analytical and transaction processing techniques together as needed to accomplish the business task.



Motivation

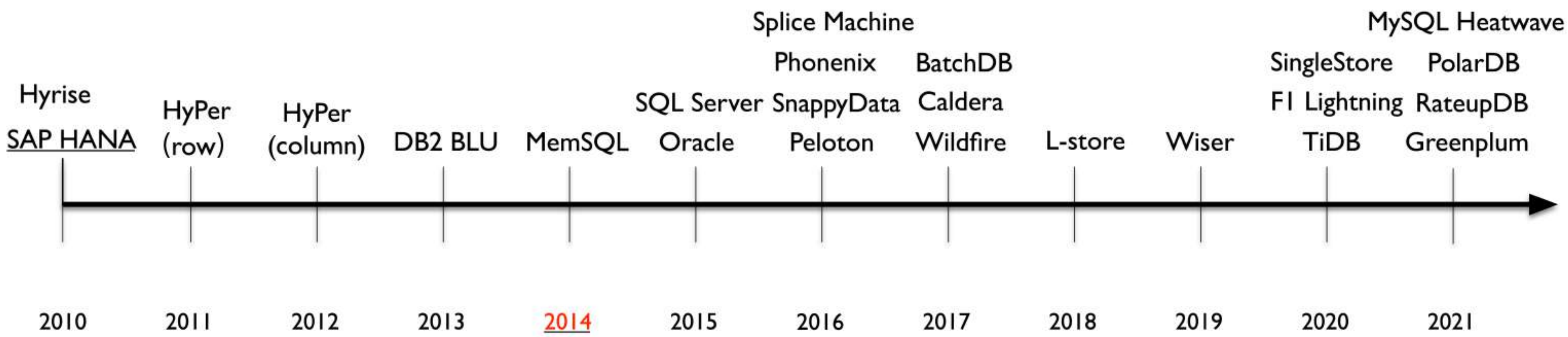
- **Gartner envisioned that, HTAP techniques will be widely adopted in the business applications with real-time data analytics by 2024.**

- **HTAP databases have many applications in E-commerce, Finance and Banking, Fraud Detection, etc.**

- **For example, identify the sales trend on-the-fly in e-commerce; detecting the fraudulent transactions when processing the transactions.**

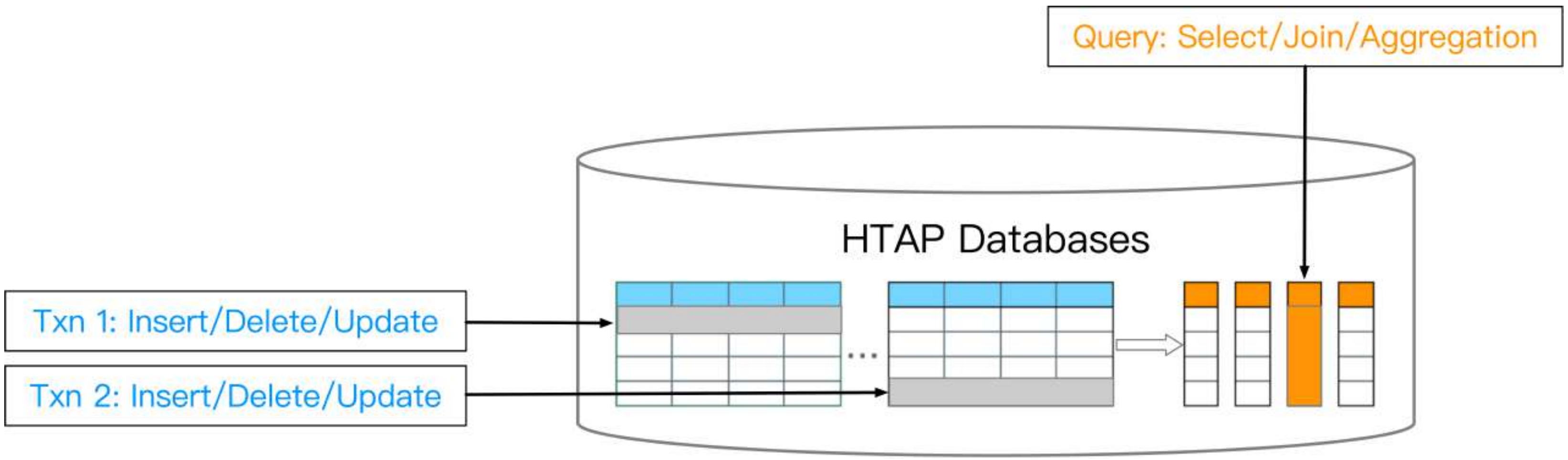
Motivation

- Over the last decade, many HTAP databases have emerged
- The following timeline consists of three phases:
  - Phase 1 (2010-2014): HTAP databases mainly adopt primary column store
  - Phase 2 (2014-2020): HTAP databases mainly extend the primary row store
  - Phase 3 (2020-present): HTAP databases utilize a distributed architecture



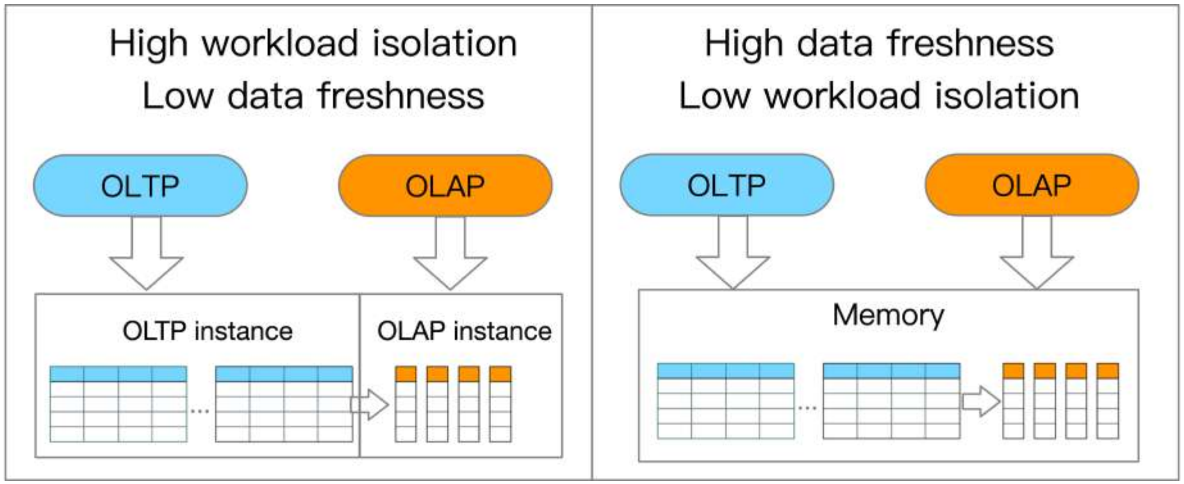
Motivation

- Rule of thumb 1: row store is ideal for OLTP workloads
  - Row-wise, update-heavy, short-lived transactions
- Rule of thumb 2: column store is best suited for OLAP workloads
  - Column-wise, read-heavy, bandwidth-intensive queries
- We study HTAP databases with both row store and column store



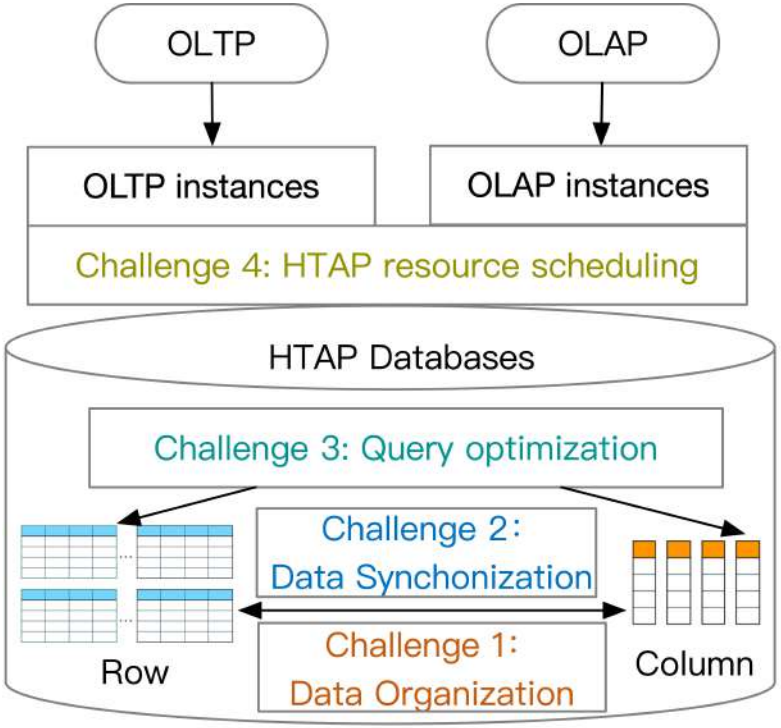
A trade-off for HTAP databases

- Workload isolation: the isolation level of handling the mixed workloads
- Data freshness: the portion of latest transaction data that is read by OLAP
- Trade-off for workload isolation and data freshness
  - High workload isolation leads to low data freshness
  - Low workload isolation results in high data freshness



Challenges for HTAP databases

- **Challenge 1 (Data Organization): how to organize the data adaptively for HTAP workloads with high performance and low storage cost.**
- **Challenge 2 (Data Synchronization): how to synchronize the data from the row store to the column store for high throughput and data freshness**
- **Challenge 3 (Query Optimization): how to optimize the query with both row store and column store by exploring the huge plan space.**
- **Challenge 4 (Resource Scheduling): how to schedule the resources for OLTP and OLTP instances effectively for high throughput and data freshness.**



Outline

- HTAP Databases
- HTAP Techniques
- HTAP Benchmarks
- Challenges and Open Problems

HTAP Databases

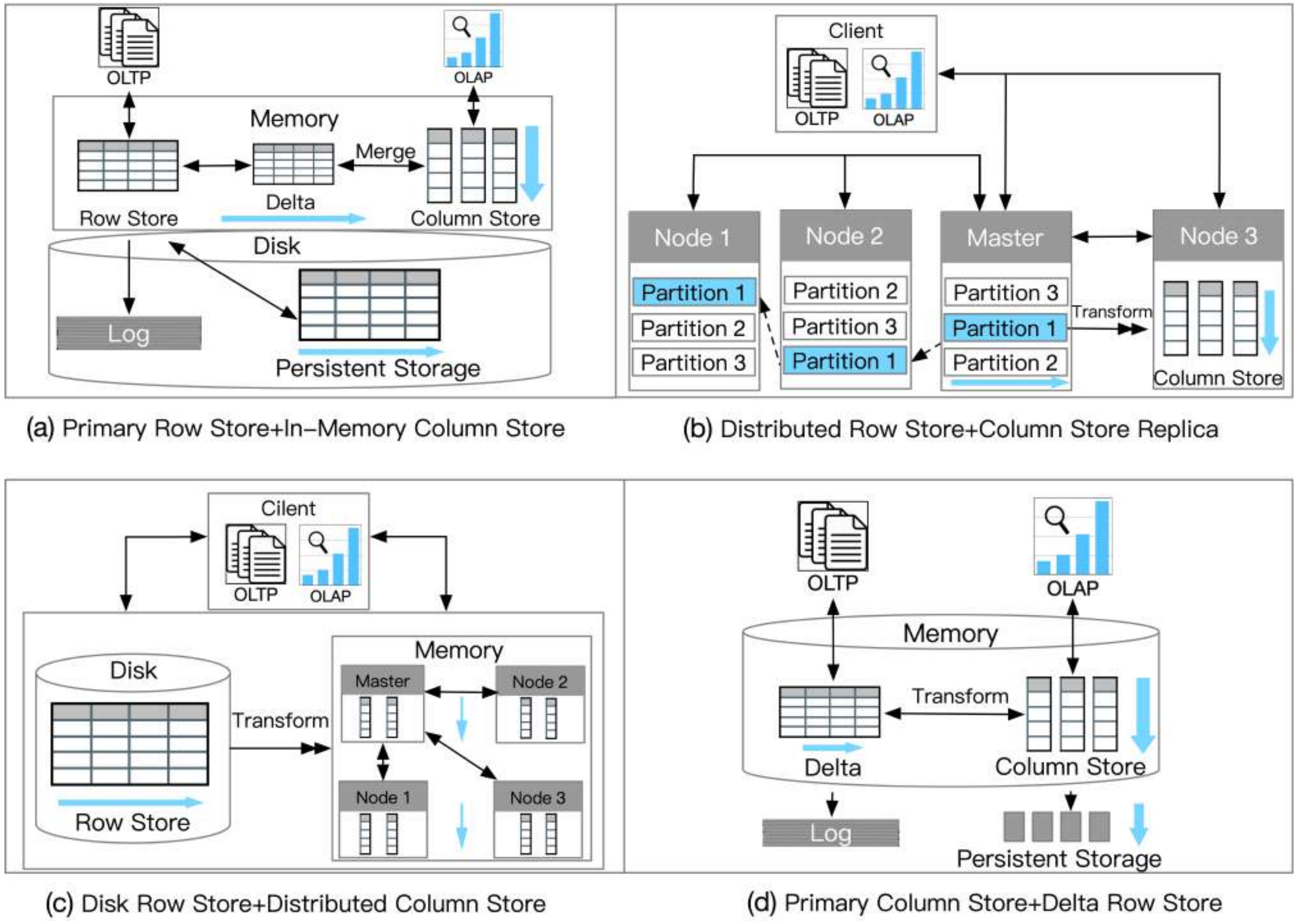
An Overview of HTAP Architectures

(a) **Primary Row Store + In-Memory Column Store**

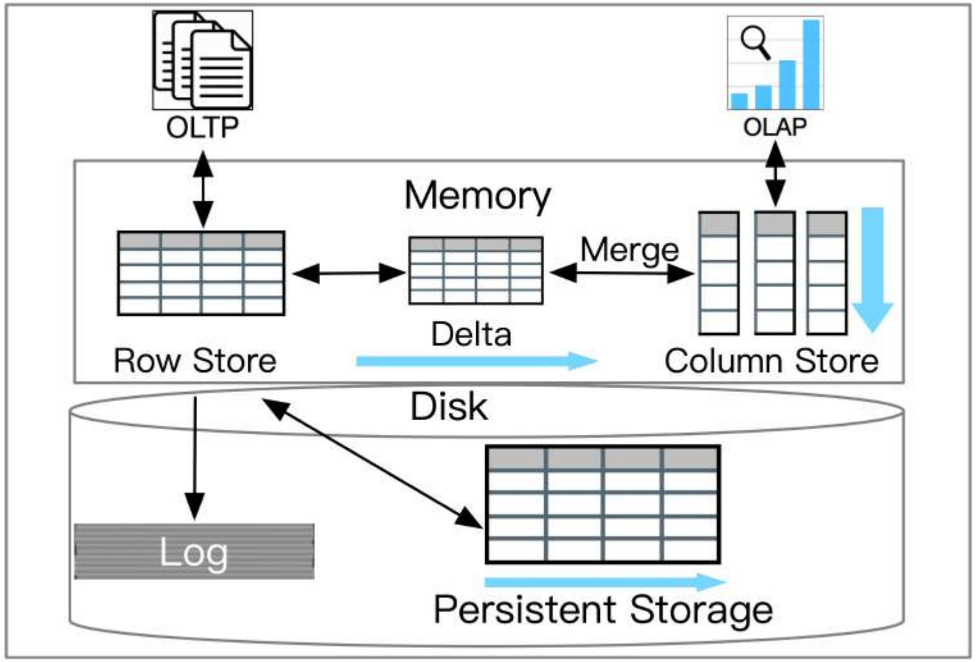
(b) **Distributed Row Store + Column Store Replica**

(c) **Disk Row Store + Distributed Column Store**

(d) **Primary Column Store + Delta Row Store**



(a) Primary Row Store+ In Memory Column Store



(a) Primary Row Store + In–Memory Column Store

Pros:

- High TP throughput,
- High AP throughput,
- High data freshness

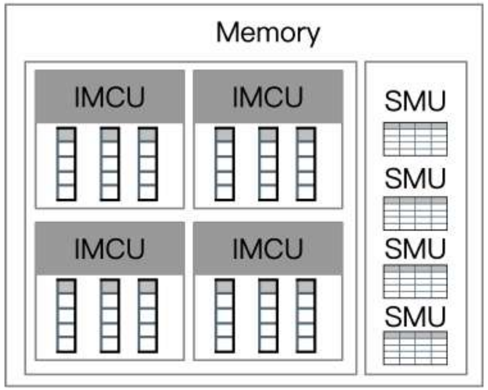
Cons:

- Low AP scalability
- Low workload isolation

Applications:

High throughput, low scalability
(e.g., banking with real-time data analytics)

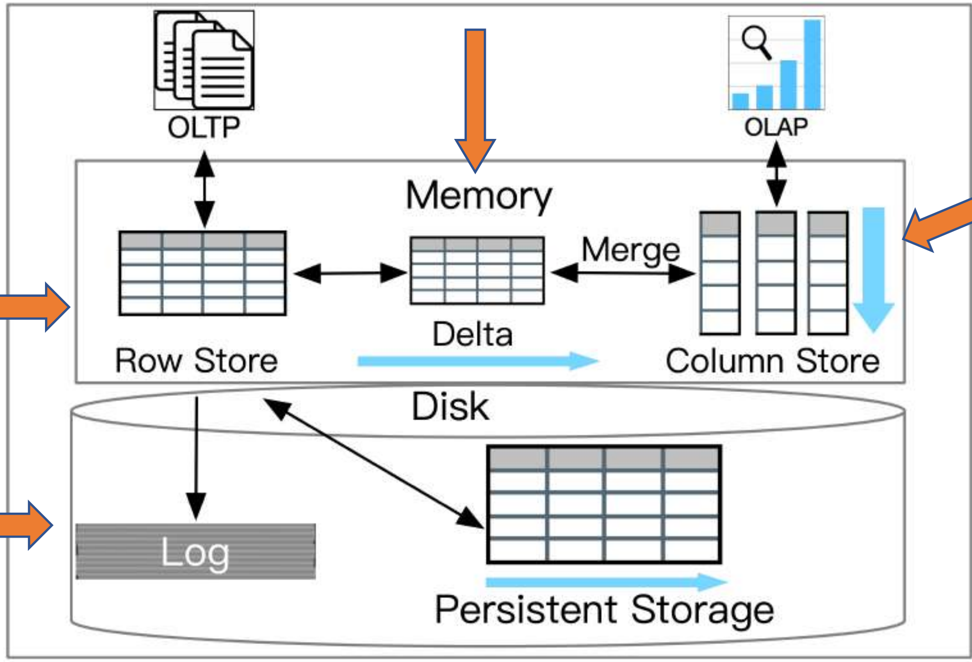
(a) Case Study: Oracle Dual-Format



Oracle Snapshot Memory Unit (SMU)

Oracle Cache

Oracle Disk-based Row Store



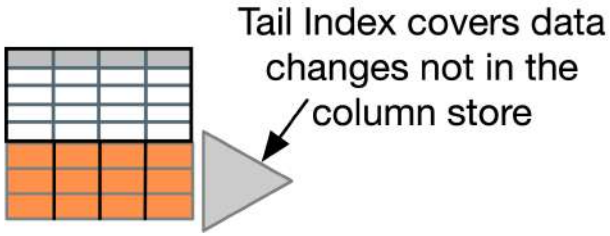
(a) Primary Row Store + In–Memory Column Store

Oracle In-memory column store (IMCS)

SIMD,
Max-Min Zone Map,
Vector Group By

Lahiri, Tirthankar, et al. "Oracle database in-memory: A dual format in-memory database." In ICDE, 2015.

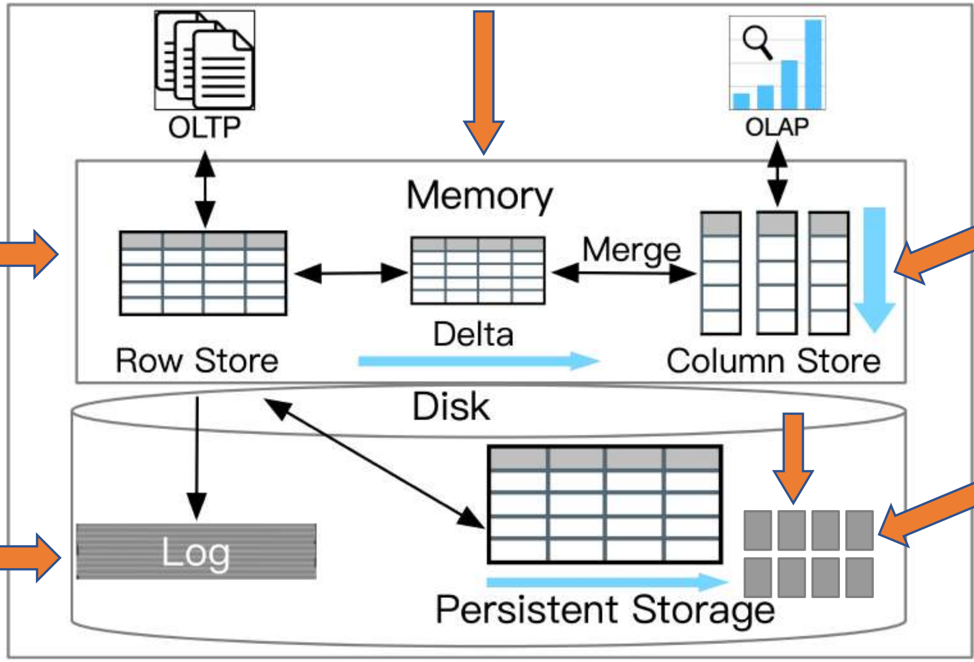
(a) Case Study: SQL Server



SQL Server Tail Index

SQL Server Hekaton

SQL Server Disk-based Row Store



(a) Primary Row Store + In–Memory Column Store

SQL Server Column Store Index (CSI)

Persistent Column Store

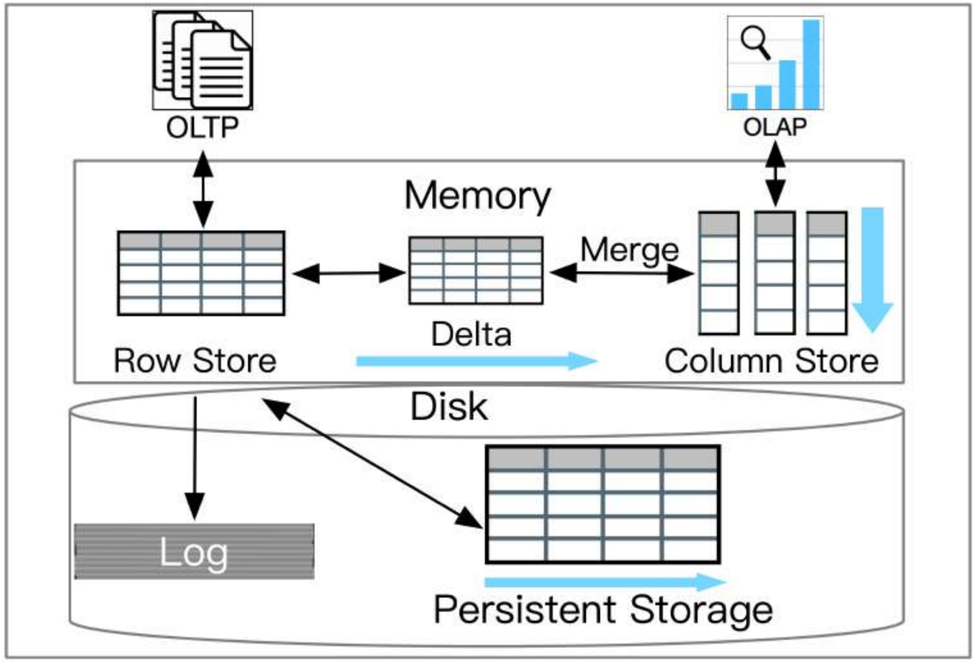
Updatable

Larson, Per-Åke, et al. "Real-time analytical processing with SQL server." *PVLDB* 8(12), 2015: 1740-1751.

Comparisons of HTAP Databases with architecture (a)

HTAP Databases	Row Store for TP	Delta	Column Store for AP	Persistent Column Store
Oracle Dual-Format	Row Store (disk) Cache (memory)	Snapshot Metadata Unit(SMU)	In Memory Column Store (IMCS)	no
SQL Server	Row Store (disk) Hekaton Row Store (memory)	Tail index	Column Store Index (CSI)	yes

Challenges for HTAP Databases with architecture (a)



(a) Primary Row Store + In–Memory Column Store

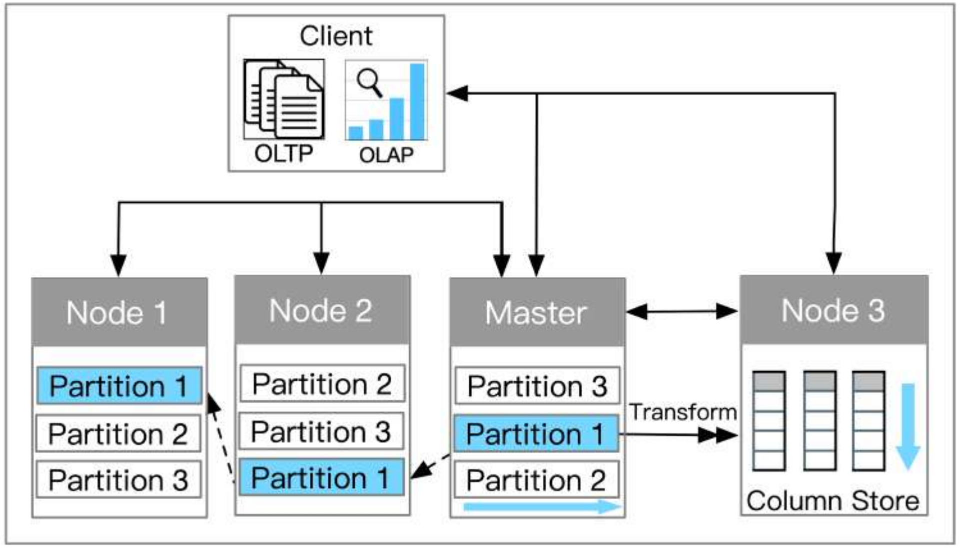
Problems: need to increase the AP scalability and workload isolation

Challenges: how to scale and isolate the AP (i.e., column store) while maintaining high TP & AP throughput and data freshness

Possible ways:

Scale up/out the memory capacity

(b) Distributed Row Store + Column Store Replica



(b) Distributed Row Store + Column Store Replica

Pros:

High workload isolation

High scalability

Cons:

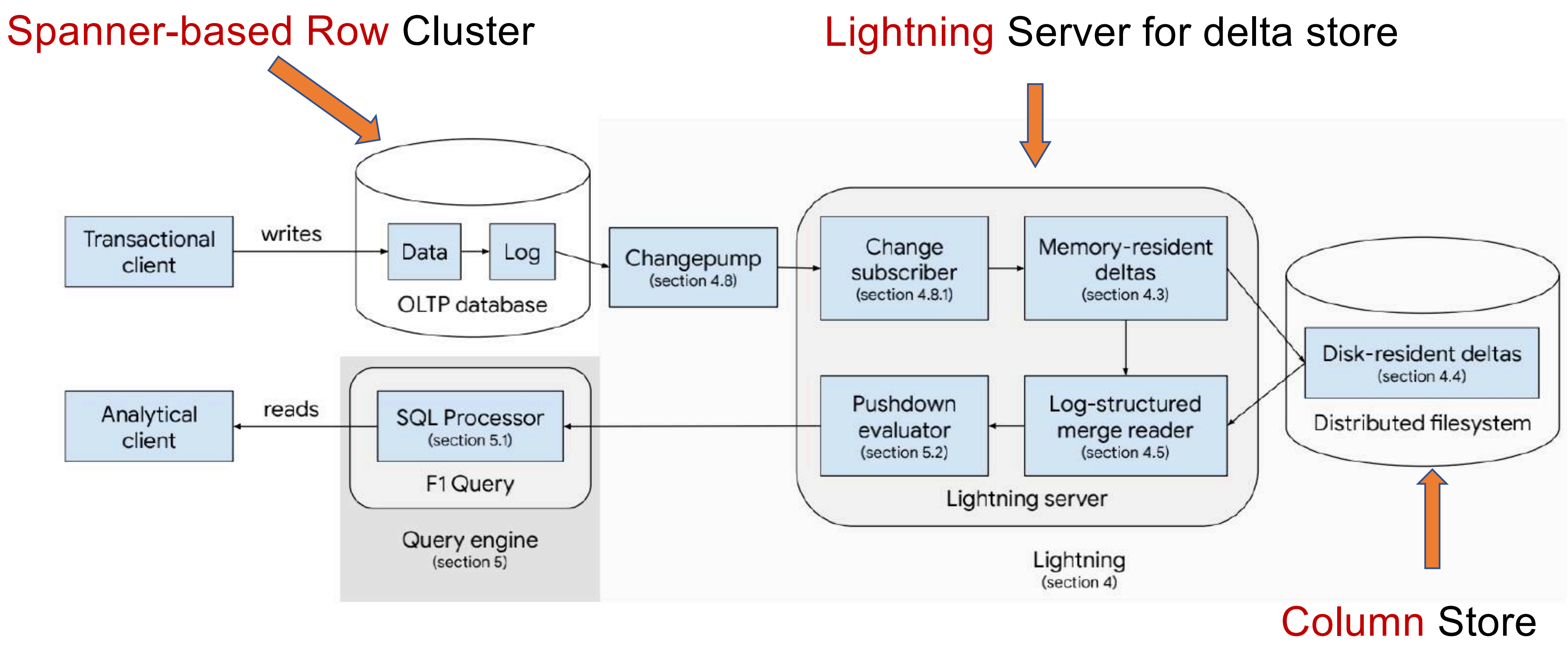
Low data freshness

Applications:

High TP&AP scalability, tolerable data freshness

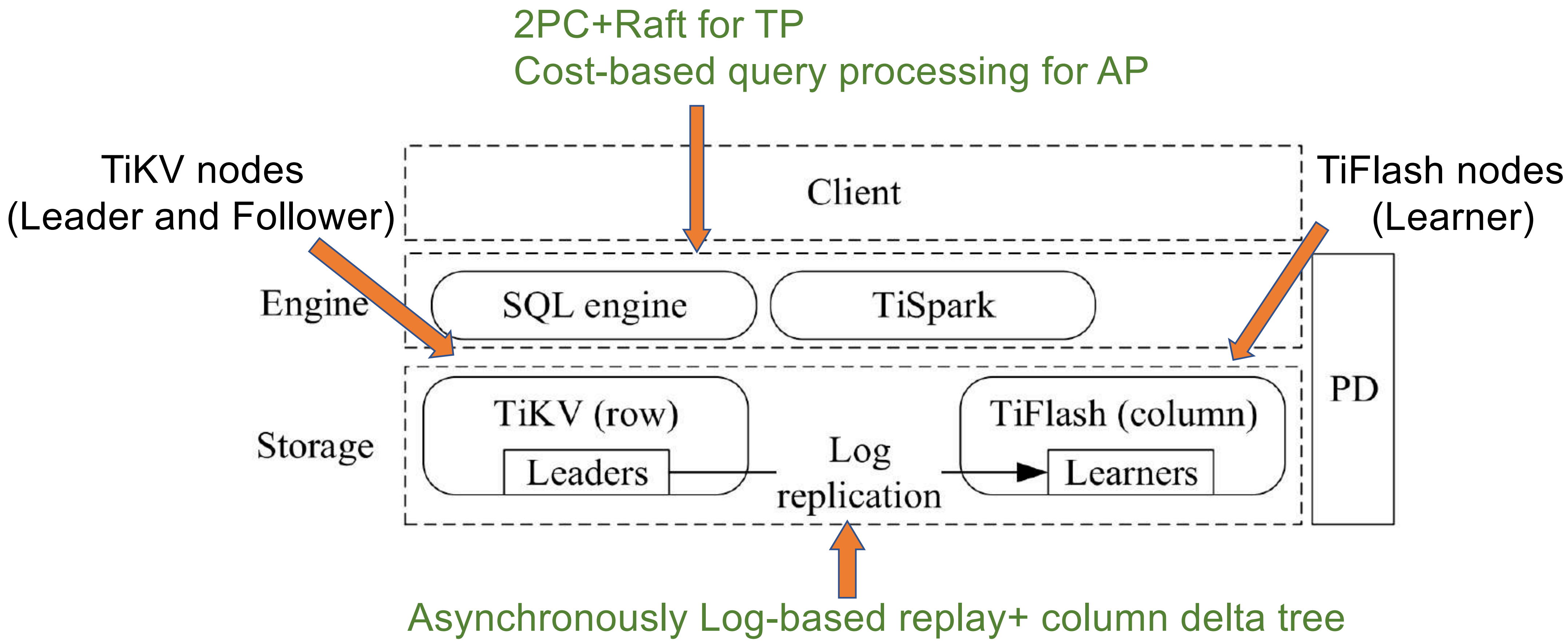
(e.g., E-commerce with real-time data analytics)

(b) Case Study: F1 Lightning



Yang, Jiacheng, et al. "F1 Lightning: HTAP as a Service." *PVLDB* 13(12), 2020: 3313-3325.

(b) Case Study: TiDB

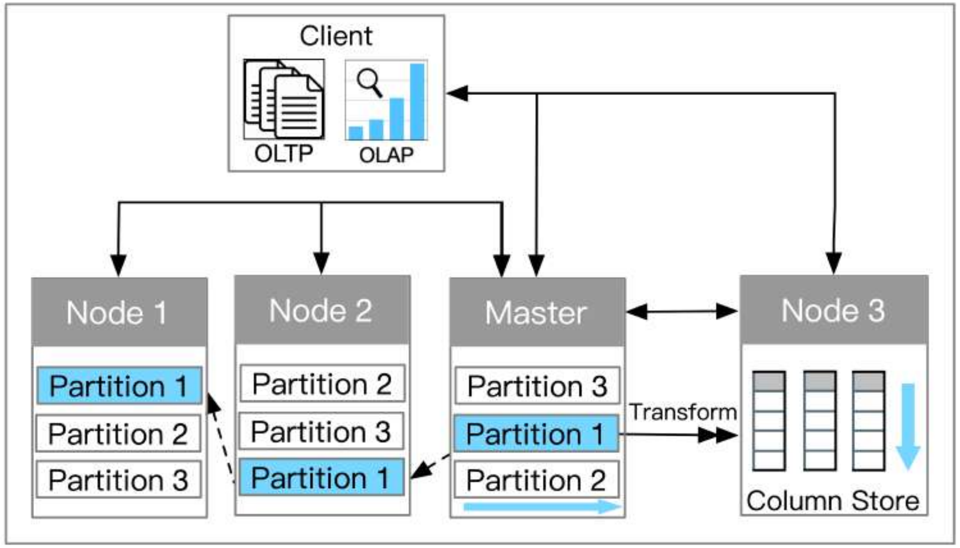


Huang, Dongxu, et al. "TiDB: a Raft-based HTAP database." *PVLDB* 13(12), 2020: 3072-3084.

Comparisons of HTAP Databases with architecture (b)

HTAP Databases	Row Store for TP	Distributed Protocol	Delta	Column Store for AP
F1 Lightning	Distributed cluster (disk)	Paxos+2PC	Log-Structured Merge tree	Distributed column store (disk)
TiDB	Distributed cluster (disk)	Raft+2PC	B-tree + Columnar delta tree	Distributed column store (disk)

Challenges for HTAP Databases with architecture (b)



(b) Distributed Row Store + Column Store Replica

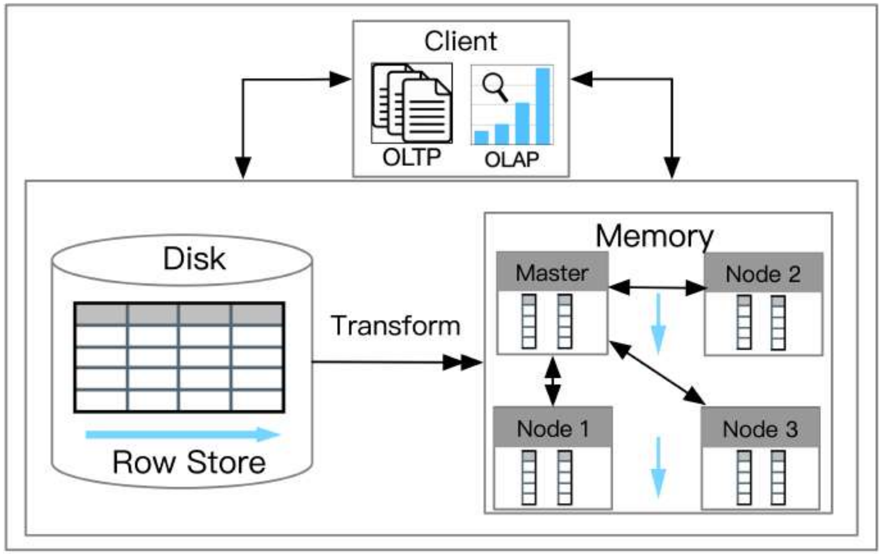
Problems: need to increase the data freshness

Challenges: how to efficiently merge the delta files to the column store

Possible solutions:

(1) Memory-based delta logging and shipping
(2) New indexing techniques for delta merging

(c) Disk Row Store + Distributed Column Store



(c) Disk Row Store + Distributed Column Store

Pros:

- High workload isolation
- High AP throughput and scalability

Cons:

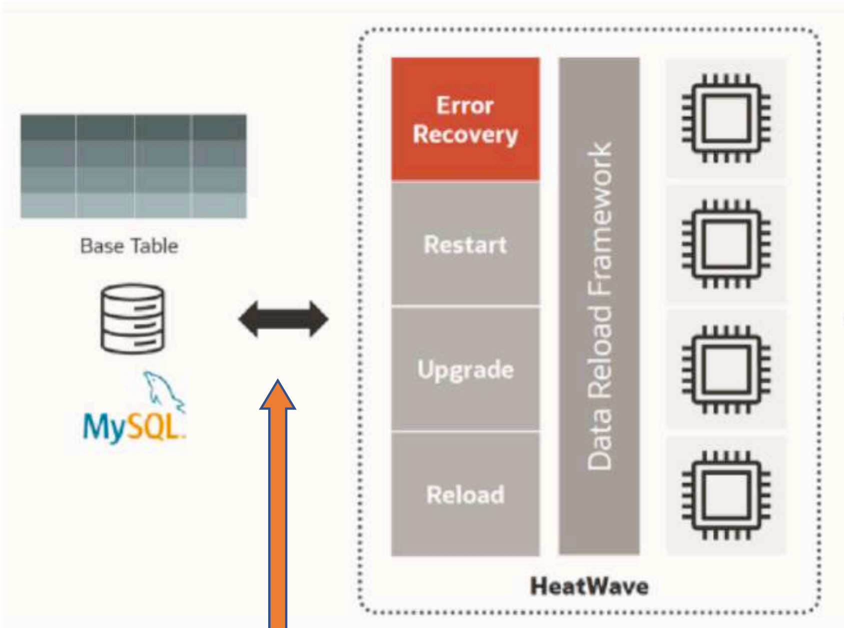
- Medium(On-premise) /Low(Cloud-based) data freshness

Applications:

High AP scalability, tolerable data freshness (e.g., IoT applications with real-time data analytics)

(c) Case Study: MySQL Heatwave

MySQL RDBMS
(Row)



Heatwave Cluster
(Column)

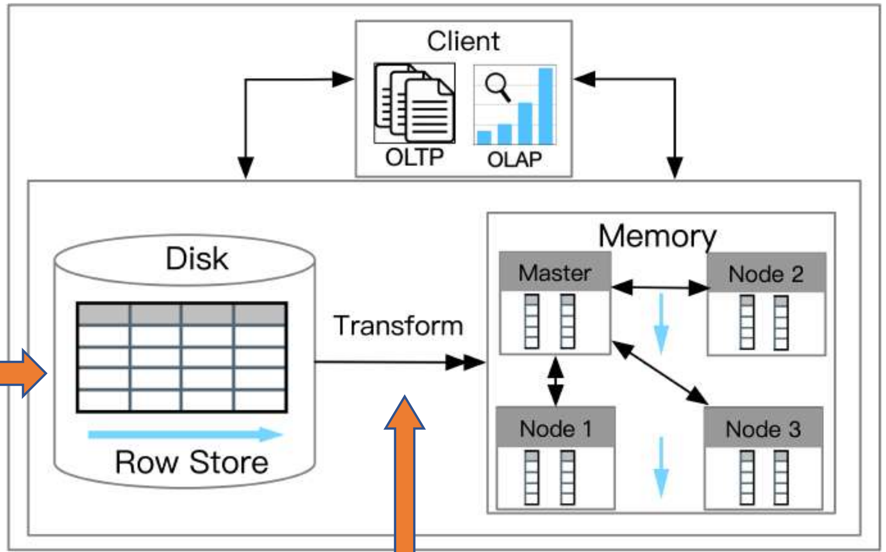
Auto-pilot service
(Data Partition,
Query Execution,
Scheduling, etc.)

Auto-Sync (every 200ms/ 64MB/
querying updated data)

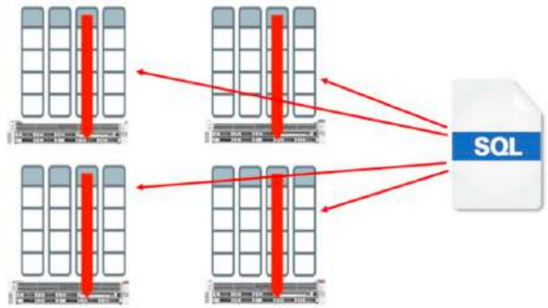
MySQL Heatwave. Real-time Analytics for MySQL Database Service, August 2021, Version 3.0

(c) Case Study: Oracle RAC

Oracle RDBMS
(Row)



(c) Disk Row Store | Distributed Column Store



RAC Cluster
(Column)

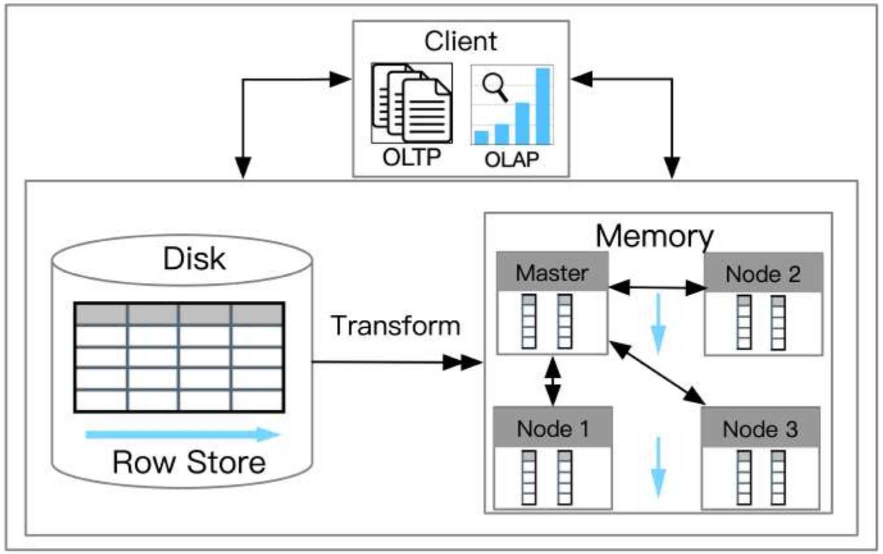
Auto-Sync (threshold-based
change propagation)

Lahiri, Tirthankar, et al. "Oracle database in-memory: A dual format in-memory database." In ICDE, 2015.

Comparisons of HTAP Databases with architecture (c)

HTAP Databases	Row Store for TP	Delta	Column Store for AP	Persistent Column Store	Automatic Column Store Management
MySQL Heatwave	MySQL RDBMS (disk)	Buffer	Distributed column store (memory)	yes	Yes (Data Partition, Query Execution, Scheduling, etc.)
Oracle RAC	Oracle RDBMS (disk)	Snapshot Metadata Unit (SMU)	Distributed column store (memory)	no	Yes (Data Load and Eviction, Query Execution, etc.)

Challenges for HTAP Databases with architecture (c)



(c) Disk Row Store + Distributed Column Store

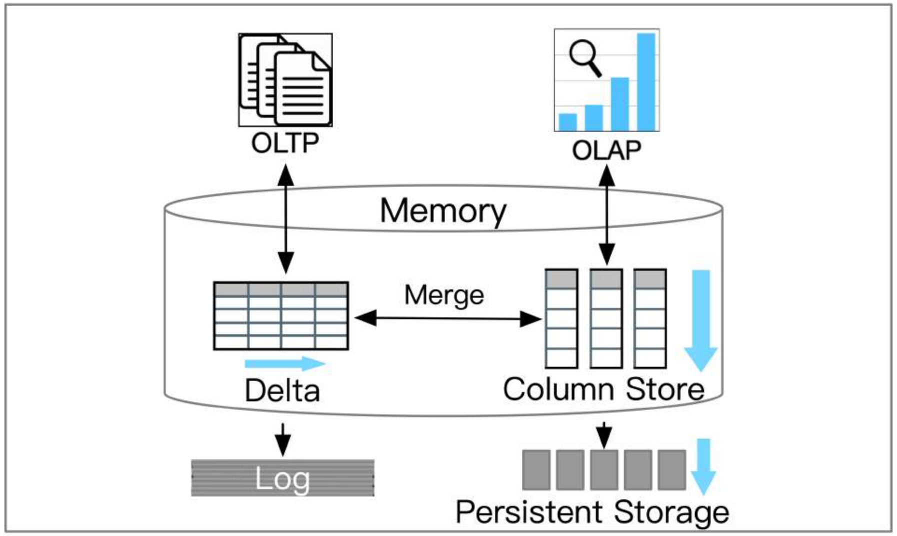
Problems: need to increase the data freshness and reduce the storage cost

Challenges: how to balance the data freshness AP throughput, and storage cost adaptively

Possible solutions:

(1) Cost models for column data management
(2) ML models for column data management

(d) Primary Column Store + Delta Row Store



(d) Primary Column Store + Delta Row Store

Pros:

High data freshness
High AP throughput

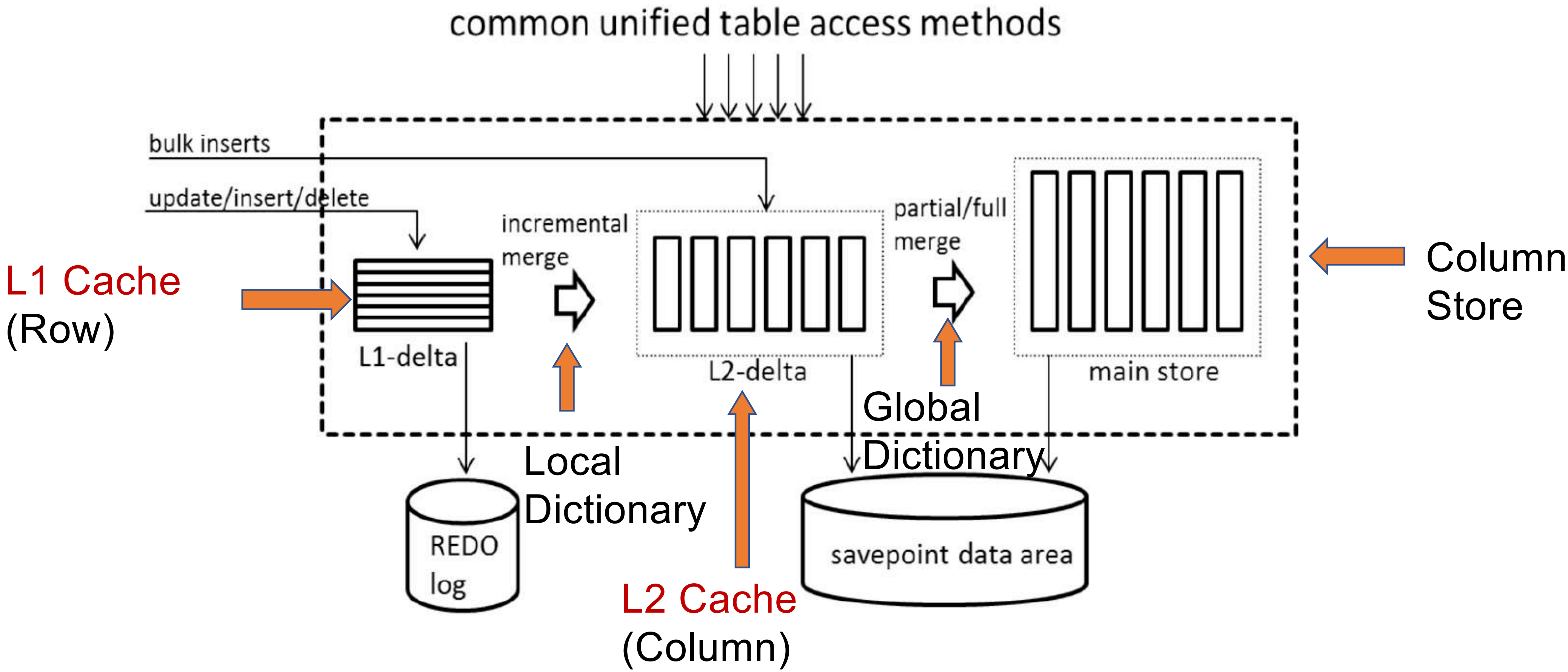
Cons:

Low TP scalability
Low workload isolation

Applications:

High AP throught, High data freshness
(e.g., Real-time Fraud Detection)

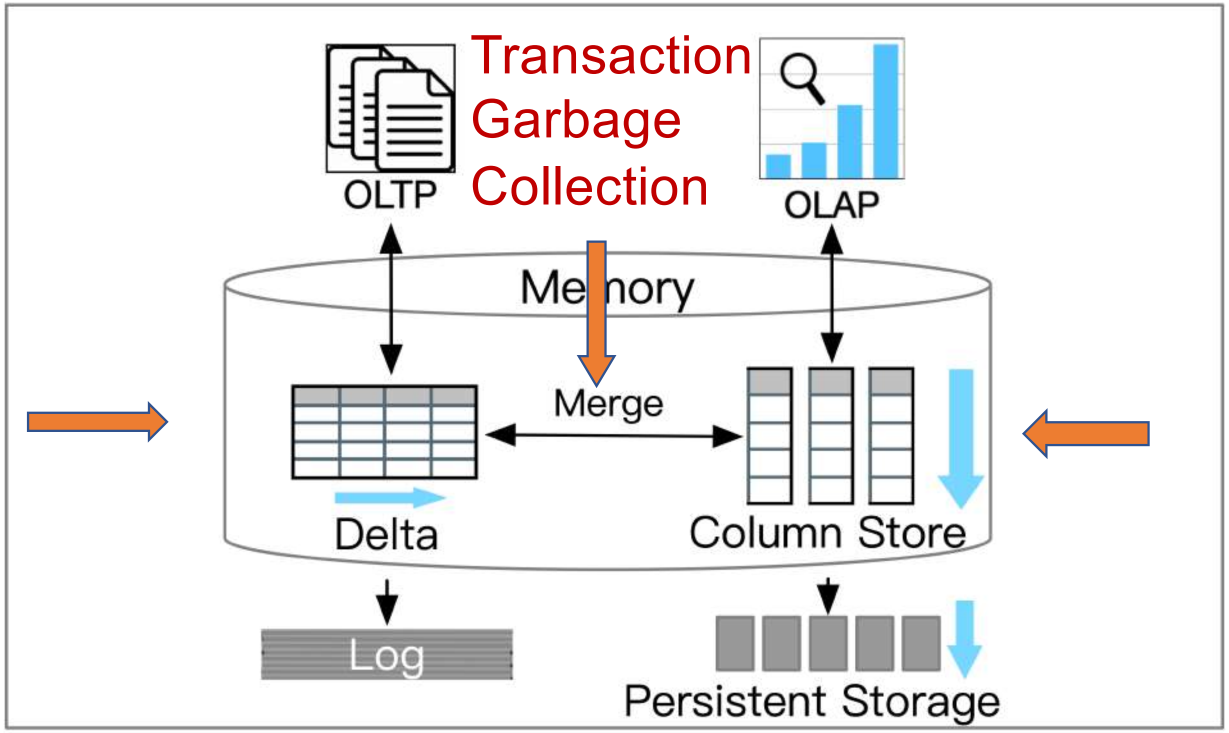
(d) Case Study: SAP HANA



Sikka, Vishal, et al. "Efficient transaction processing in SAP HANA database: the end of a column store myth." In *SIGMOD*. 2012.

(d) Case Study: Hyper (Column)

Version Vector



(d) Primary Column Store + Delta Row Store

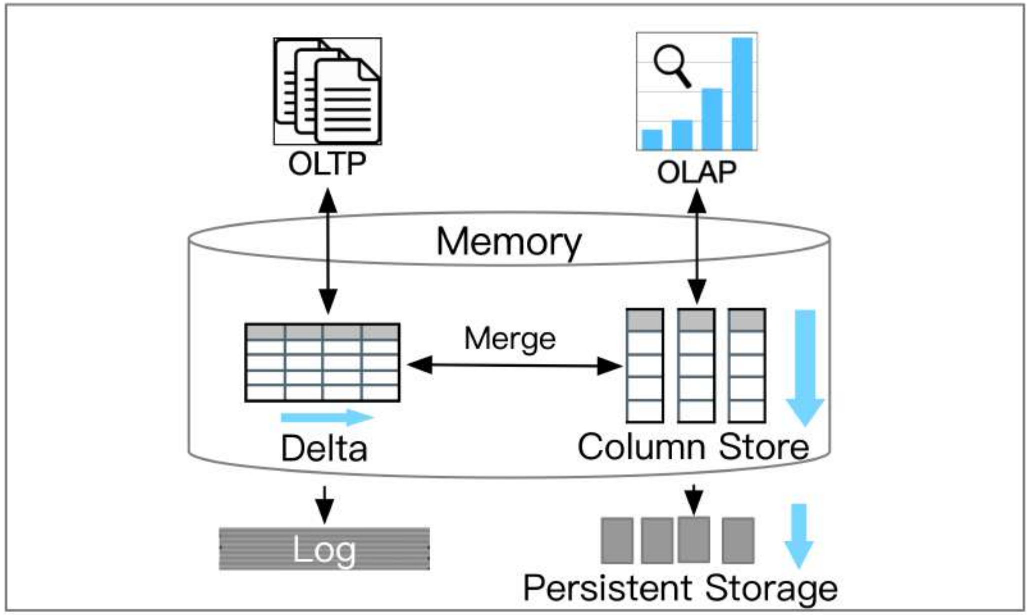
Column Store

Neumann, Thomas, Tobias Mühlbauer, and Alfons Kemper. "Fast serializable multi-version concurrency control for main-memory database systems." *In SIGMOD* ,2015.

Comparisons of HTAP Databases with architecture (d)

HTAP Databases	Row Store for TP	Delta	Column Store for AP	Data Synchronization
SAP HANA	L1 Cache	L2 Cache	In-Memory Column Store	Dictionary-based merging
Hyper	Buffer	Version Vector	In-Memory Column Store	Transaction-Level Garbage Collection

Challenges for HTAP Databases with architecture (d)



(d) Primary Column Store + Delta Row Store

Problems:

1. need to increase the TP scalability

2. need to increase workload isolation

Challenge: how to traverse the delta storage efficiently while keeping high throughput for HTAP

Possible solutions:

(1) Trade data freshness for AP throughput
(2) New Indexing techniques for delta traversal and delta merging

A summary of HTAP databases

Category	HTAP Databases	OLTP Throughput	OLAP Throughput	OLTP Scalability	OLAP Scalability	Workload Isolation	Data Freshness
Primary Row Store+ In Memory Column Store	Oracle Dual-Format SQL Server, DB2 BLU	High	High	Medium	Low	Low	High
Distributed Row Store + Column Store Replica	TiDB, F1 Lightning SingleStore	Medium	Medium	High	High	High	Low
Disk Row Store + Distributed Column Store	MySQL Heatwave, Oracle RAC	Medium	Medium	Medium	High	High	Medium
Primary Column Store + Delta Row Store	SAP HANA (without scale-out), Hyper	Medium	High	Low	Medium	Low	High

Other HTAP systems

Other categories of HTAP systems

Row-based HTAP systems

- ❑ Hyper (Row): support HTAP with copy-on-write mechanism

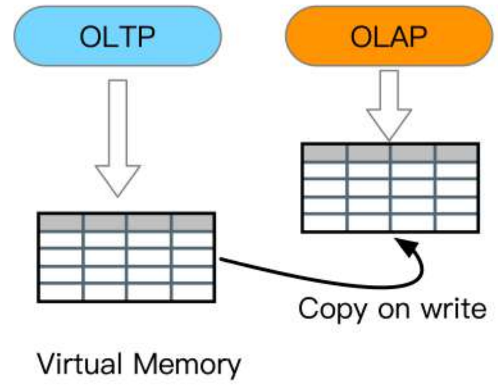
  Pros: High data freshness, High TP throughput

  Cons: Low workload isolation, Low AP throughput & Scalability

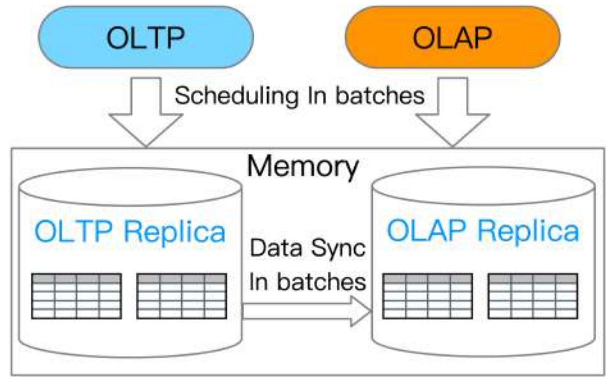
- ❑ BatchDB: row-based dual-store with batched workload scheduling

  Pros: High TP throughput

  Cons: Low data freshness, Low AP throughput



Kemper, Alfons, and Thomas Neumann. "HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots." *In ICDE*, 2011.



Makreshanski, Darko, et al. "BatchDB: Efficient isolated execution of hybrid OLTP+ OLAP workloads for interactive applications." *In SIGMOD,* 2017.

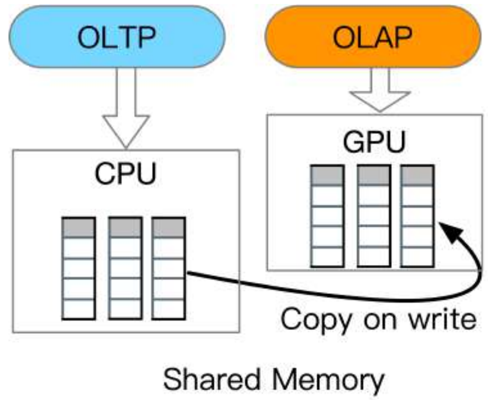
Other categories of HTAP systems

Column-based HTAP systems

- ❑ Caldera: copy-on-write column store with CPU/GPU architecture

  Pros: High data freshness, High AP throughput

  Cons: Low workload isolation, Low TP throughput

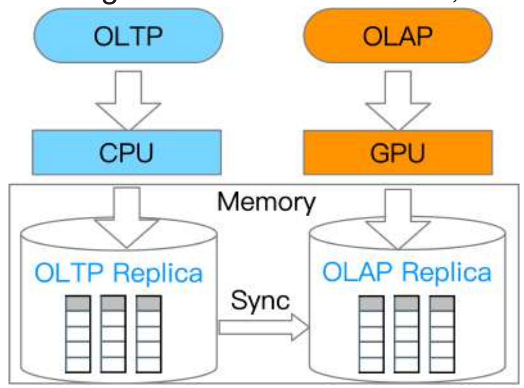


Appuswamy, Raja, et al. "The case for heterogeneous HTAP." *In CIDR,* 2017.

- ❑ RateupDB: column-based dual-store with CPU/GPU architecture

  Pros: High AP throughput

  Cons: Low data freshness, Low TP throughput



Lee, Rubao, et al. "The art of balance: a RateupDB™ experience of building a CPU/GPU hybrid database product." *PVLDB* 14(12), 2021: 2999-3013.

Other categories of HTAP systems

Spark-based HTAP systems

- ❑ Splice Machine: loosely couple HBase (TP) with Spark (AP)

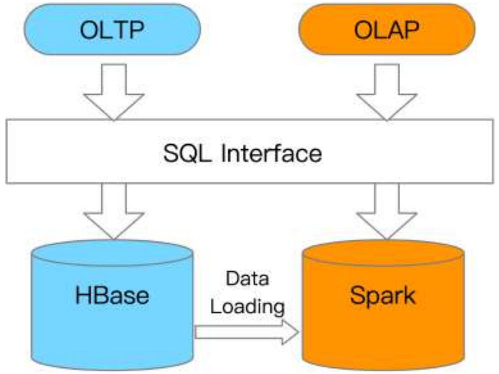
  Pros: High TP/AP scalability, High workload isolation

  Cons: Low data freshness

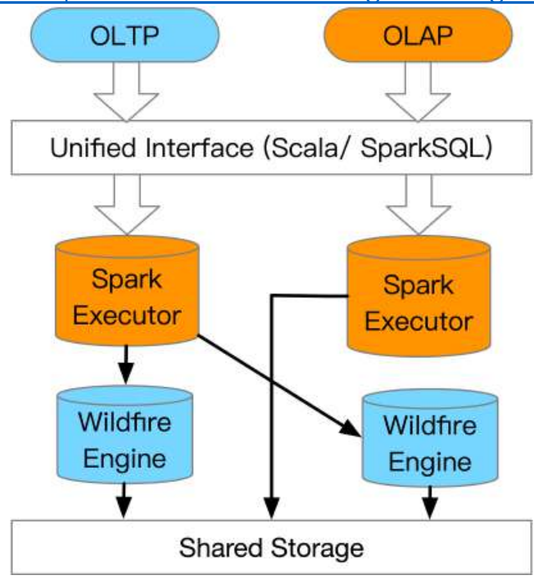
- ❑ Wildfire: tightly couple OLTP engines with Spark (AP)

  Pros: High TP/AP scalability

  Cons: Low data freshness



https://splicemachine.com/blog/defining-htap/

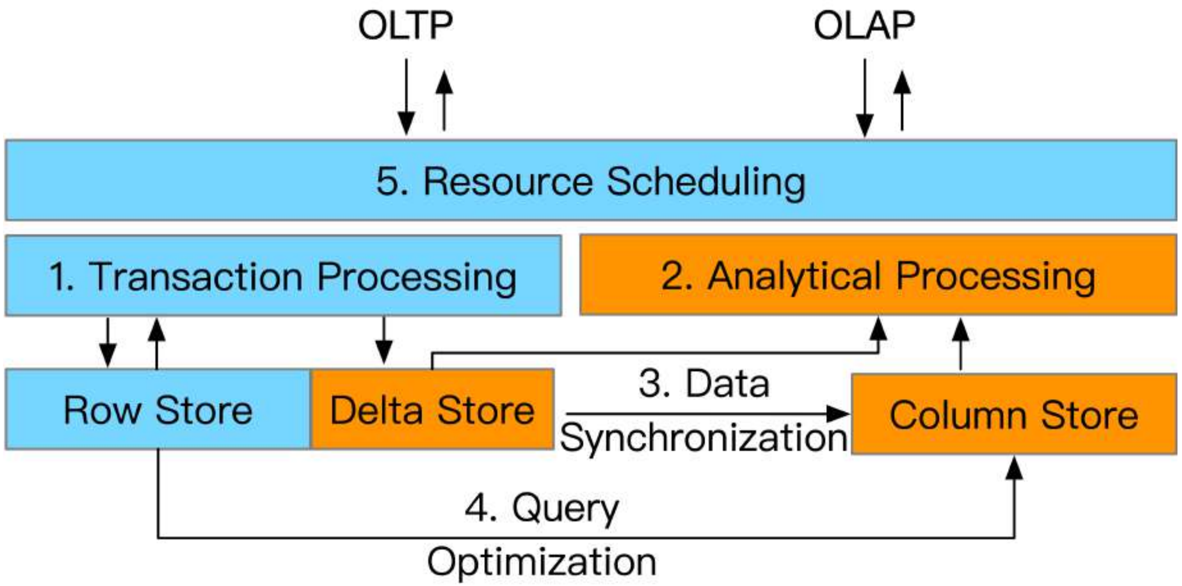


Barber, Ronald, et al. "Evolving Databases for New-Gen Big Data Applications." *CIDR*. 2017.

HTAP Techniques

Overview of HTAP Techniques

1. **Transaction Processing:** updating the **row store** and writing the **delta store**
2. **Analytical Processing:** scanning the **column store** with **delta store**
3. **Data Synchronization:** merging the **delta data** to **column store**
4. **Query Optimization:** planning queries against **row store** and **column store**
5. **Resource Scheduling:** scheduling resources for **OLTP** and **OLAP** instances



Transaction Processing

1. Standalone Transaction Processing with In-Memory Delta Update

- ❑ Standalone transaction processing with MVCC protocol
- ❑ In-memory delta update for insert/delete/update operations
- ❑ E.g., Oracle, SQL Server, SAP HANA

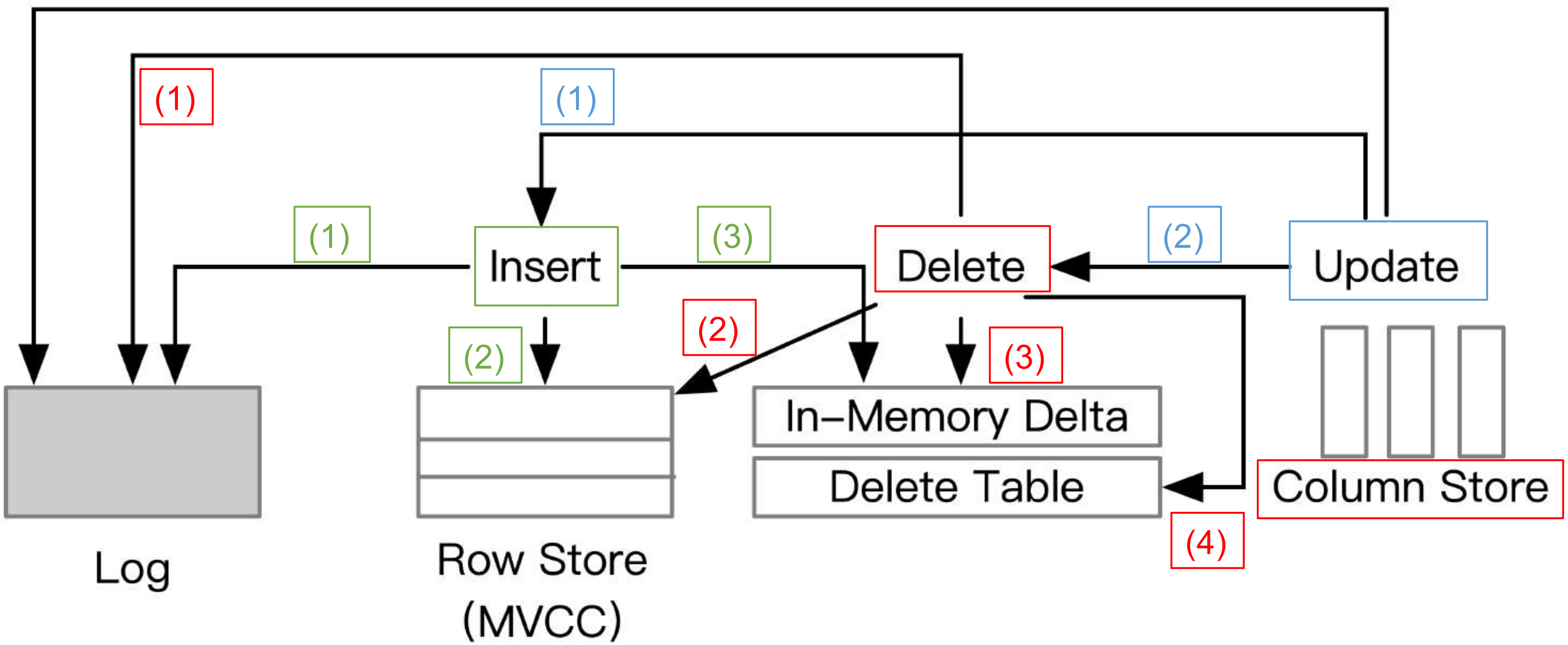
2. Distributed Transaction Processing with Log Replay

- ❑ Two-phase commit (2PC)+Paxos for distributed TP and data replication
- ❑ Log replay for updating the row store and column store
- ❑ E.g., F1 Lightning, TiDB

Master-slave replication for distributed TP, e.g., Singlestore

Transaction Processing

1. **Standalone TP for insert/delete/update operations**



Transaction Processing

- **Three implementations for an in-memory delta store:**

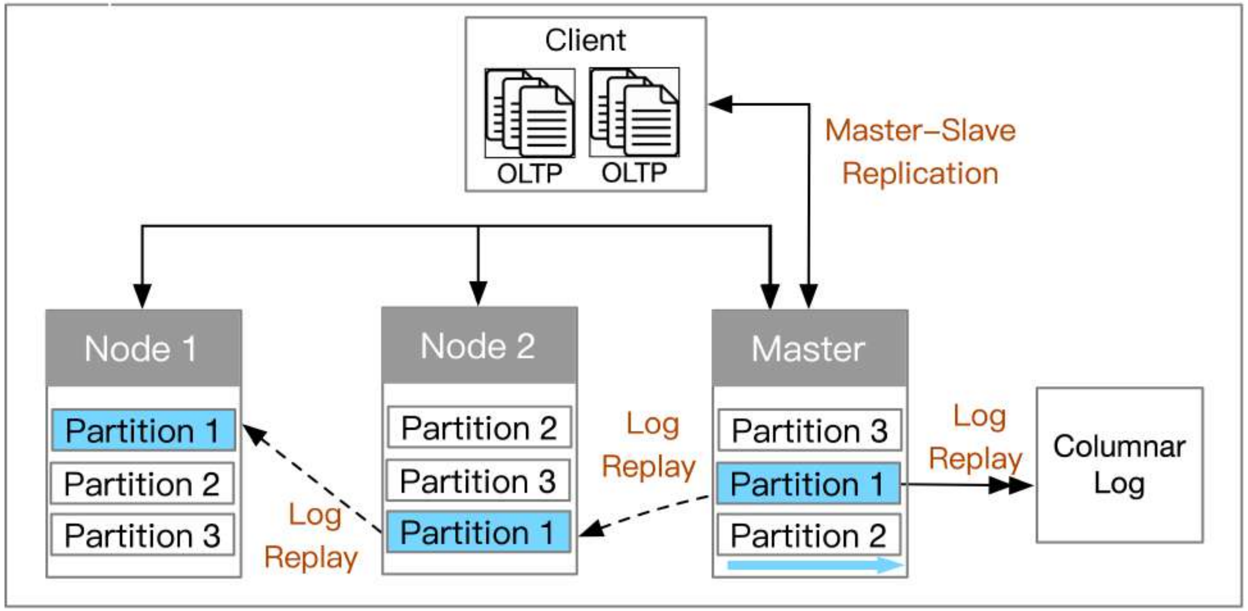
  (1) Heap table, (2) Index organized table, (3) L1 cache

Delta Store	Databases	Pros	Cons
Heap table	Oracle	Fast Insertion	Slow lookup
Index Organized Table	SQL Server	Fast lookup	Slow Insertion
L1 Cache	SAP HANA	Fast Insertion	Slow lookup Low Capacity

Transaction Processing

2. Distributed Transaction Processing with Log Replay

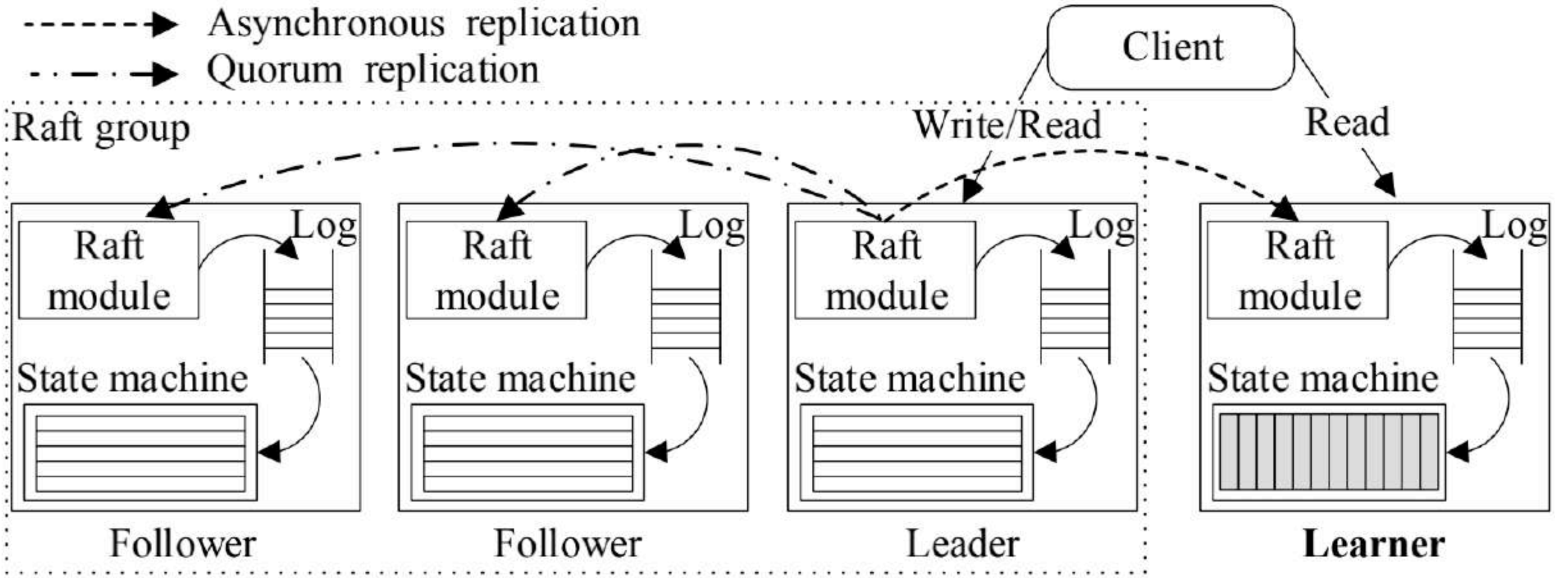
- ❑ Master-slave replication
- ❑ Master node handles the transactions, then replicate the logs to slave nodes
- ❑ E.g., SingleStore



Transaction Processing

Modified Raft Protocol for TP and AP nodes

- Leader (row), Follower (row), Learner (column)



Huang, Dongxu, et al. "TiDB: a Raft-based HTAP database." *PVLDB* 13(12), 2020: 3072-3084.

Comparisons of TP techniques in HTAP Databases

Transaction Processing Type	Databases	TP techniques	Delta	Pros	Cons
Standalone TP + In-memory Delta Update	Oracle, SQL Server	MVCC	In-memory Delta	High Efficiency	Low Scalability
Distributed TP + Log Replay	SingleStore	Master-Slave Replication	Log Files	High Efficiency	Low Freshness
	TiDB, F1 Lightning	2PC+Paxos	Log Files	High Scalability	Low Efficiency

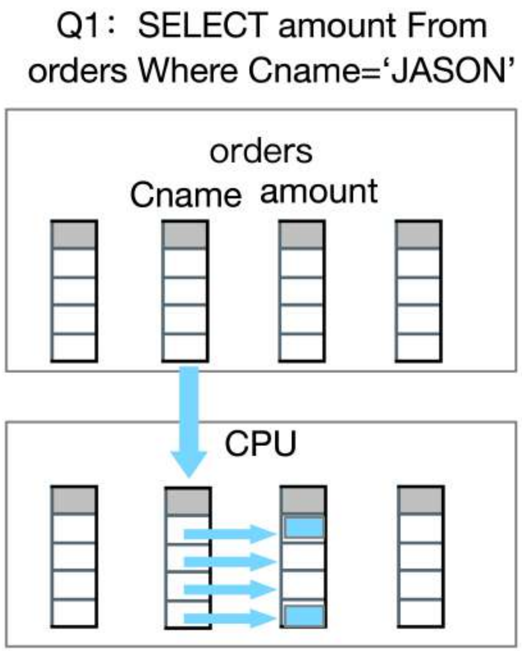
Analytical Processing

1. **Standalone Columnar Scan with In-Memory Delta Traversing**
   - ❑ Single Instructions Multiple Data (SIMD), Vector Processing
   - ❑ In-Memory Delta Traversing
   - ❑ E.g., Oracle, SQL Server

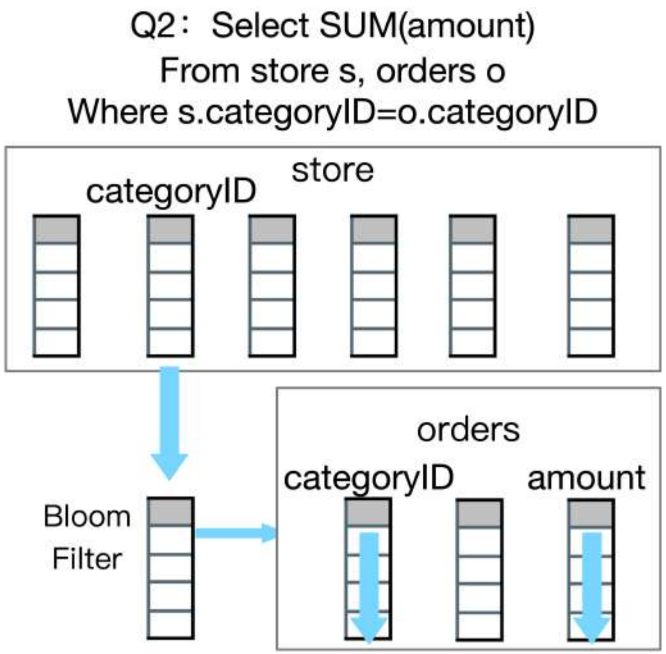
2. **Distributed Columnar Scan with Log File Scanning**
   - ❑ Distributed Query Processing over Columnar Segments
   - ❑ Disk-based Log Files Merging and Scanning
   - ❑ E.g., F1 Lightning, TiDB

Analytical Processing

1. Standalone Columnar Scan with In-Memory Delta Traversing

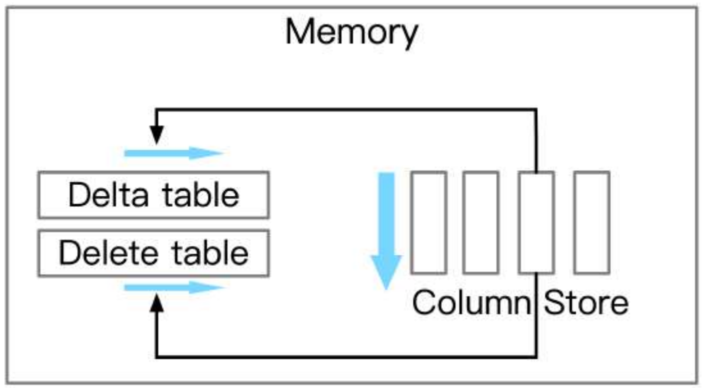


(a) SIMD query processing



(b) Vector join based on a bloom filter

Fetch visible values in the delta table and skip stale data in the delete table

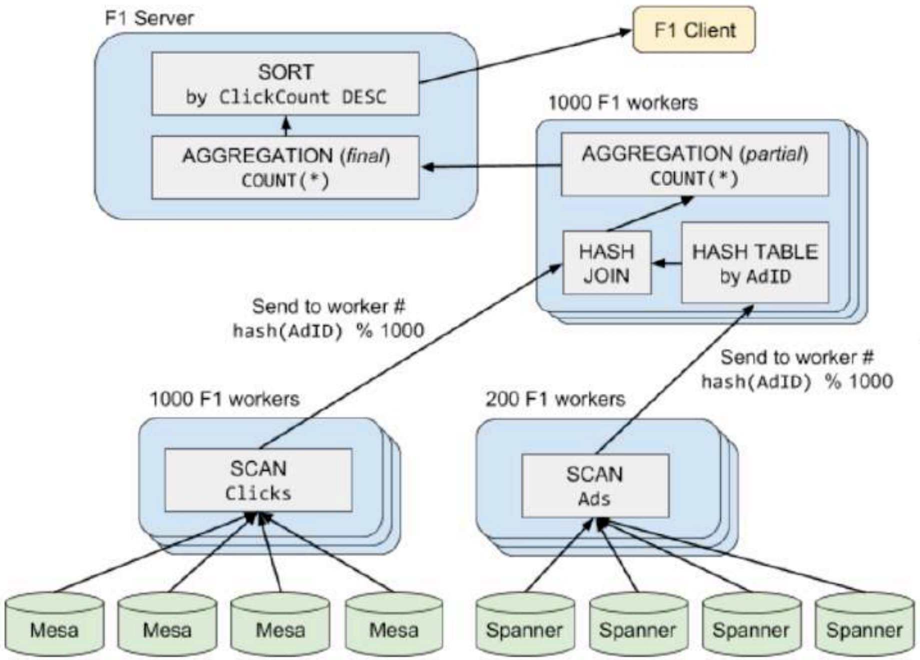


(c) Column Scan with delta traversing

Analytical Processing

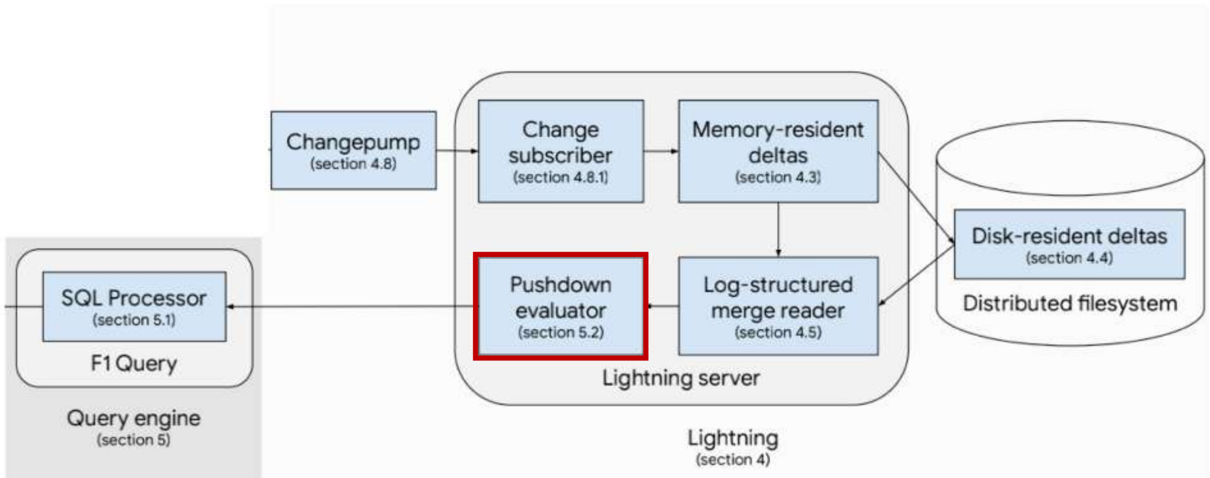
2. Distributed Columnar Scan with Log File Scanning

Distributed Columnar Scan



Samwel, Bart, et al. "F1 query: Declarative querying at scale." *PVLDB* 11(12) , 2018: 1835-1848.

Log File Scanning



Yang, Jiacheng, et al. "F1 Lightning: HTAP as a Service." *PVLDB* 13(12), 2020: 3313-3325.

Comparisons of AP techniques in HTAP Databases

Analytical Processing Type	Databases	AP techniques	Delta	Pros	Cons
Standalone Columnar Scan + In-Memory Delta Traversing	Oracle, SQL Server, SAP HANA	Vector query processing + Delta traversing	In-memory delta table	High Freshness	Large Memory Size
Distributed Columnar Scan + Log File Scanning	TiDB, F1 Lightning	Distributed query processing + Log scanning	Disk-based log files	High Scalability	Low Efficiency

Data Synchronization

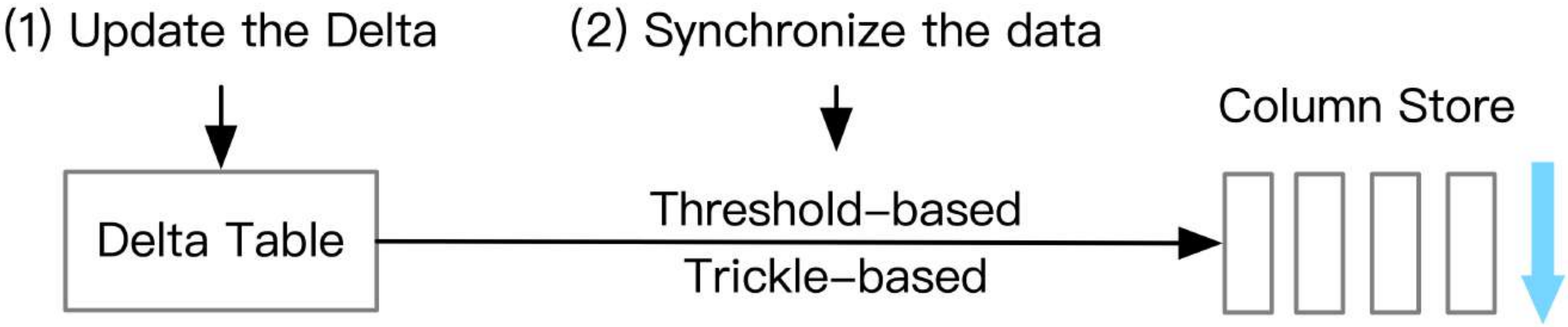
Periodically merge the latest transaction data to the column store

- ❑ **Type 1: In-memory delta merge**
  - **(1) Threshold-based merging**
  - **(2) Two-phase data migration**
  - **(3) Dictionary-based migration**
  - **Oracle, SQL Server, SAP HANA**
- ❑ **Type 2: Log-based delta merge**
  - **LSM-tree and B-tree**
  - **TiDB, F1 Lightning**

Data Synchronization

1. In-Memory Delta Merging

- ❑ Method 1: Threshold-based merging
- ❑ e.g., threshold reaches 90% of column store

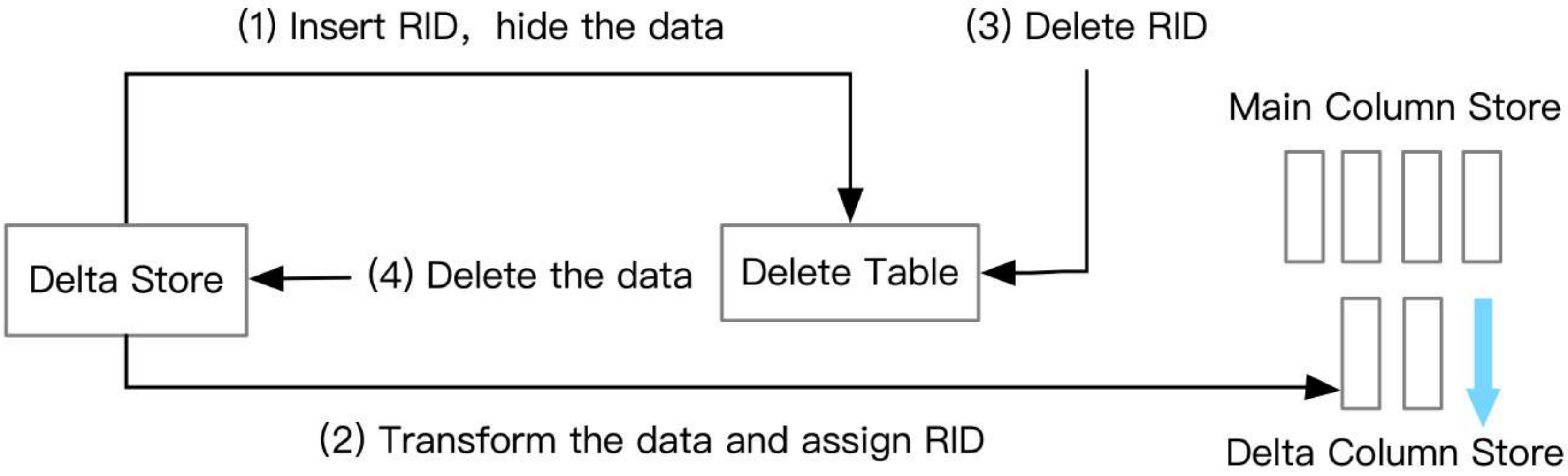


(a) Threshold-based merging

Data Synchronization

1. In-Memory Delta Merging

- ❑ Method 2: Two-phase delta migration
- ❑ Phase 1: Preparation on migration
- ❑ Phase 2: Operation on migration

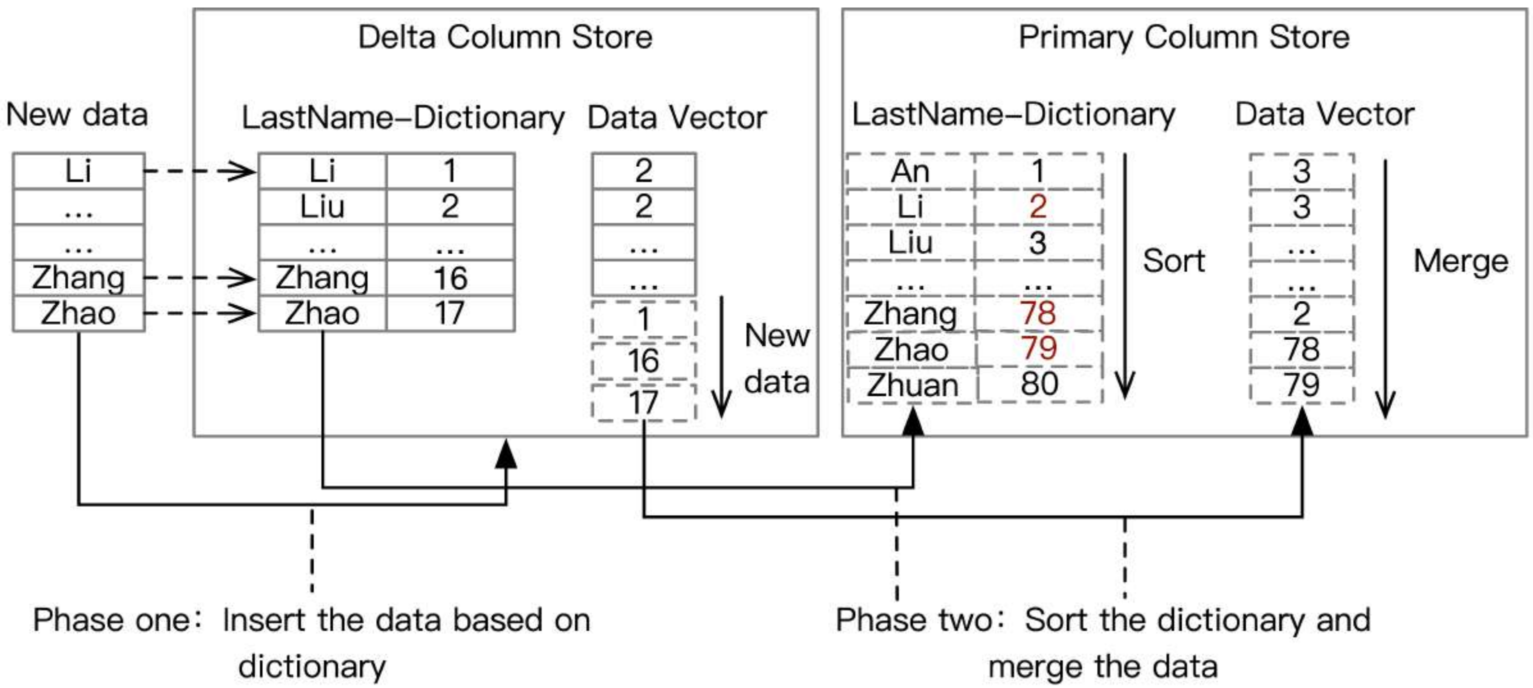


(b) Two-phase data migration

Data Synchronization

1. In-Memory Delta Merging

❑ Method 3: Dictionary-based merging

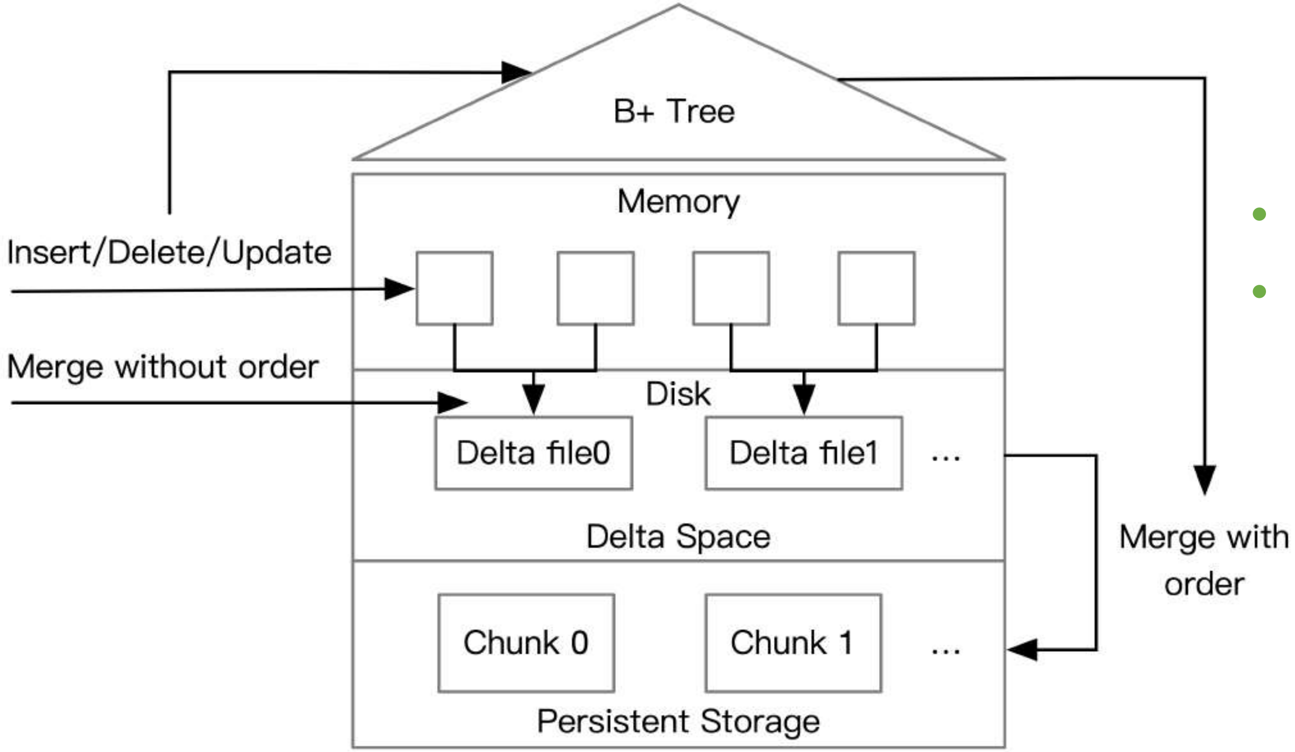


(c) Dictionary-based merging

Data Synchronization

2. Log-based delta merge

- ❑ **Memory-resident deltas (row-wise)**
- ❑ **Disk-resident deltas (column-wise)**



- Merging and Collapsing
- B+-Tree for fast merging

Comparisons of DS techniques in HTAP Databases

Data Synchronization	Databases	DS techniques	Pros	Cons
In-Memory delta merge	Oracle, SQL Server, SAP HANA	Threshold-based merging, Two-phase delta migration, Dictionary-based merging	High Efficiency	Low Scalability
Log-based delta merge	TiDB, F1 Lightning	Multi-level deltas, B+tree, log merging	High Scalability	High Merge Cost

Query Optimization in HTAP databases

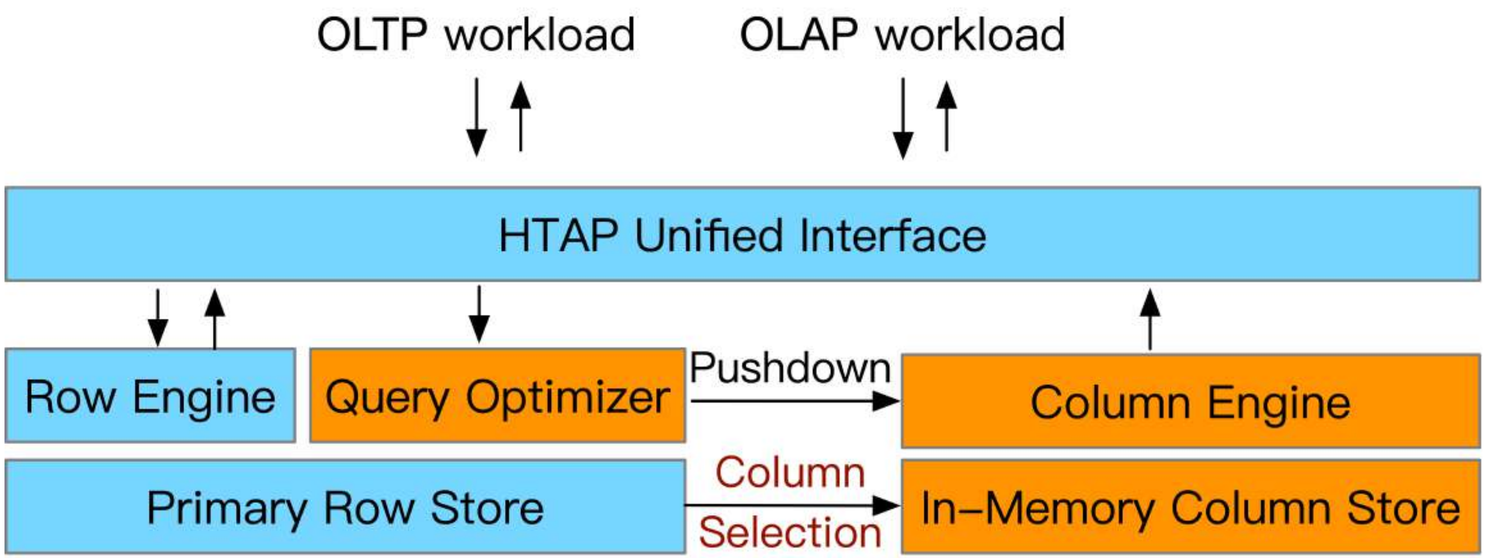
1. **In-Memory Column Selection**
   – **From a point of view of query-driven column selection**

2. **Hybrid Row and Column Scan**
   – **From a point of view of physical plan selection**

3. **CPU/GPU Acceleration for HTAP**
   – **From a point of view of hardware acceleration for HTAP**

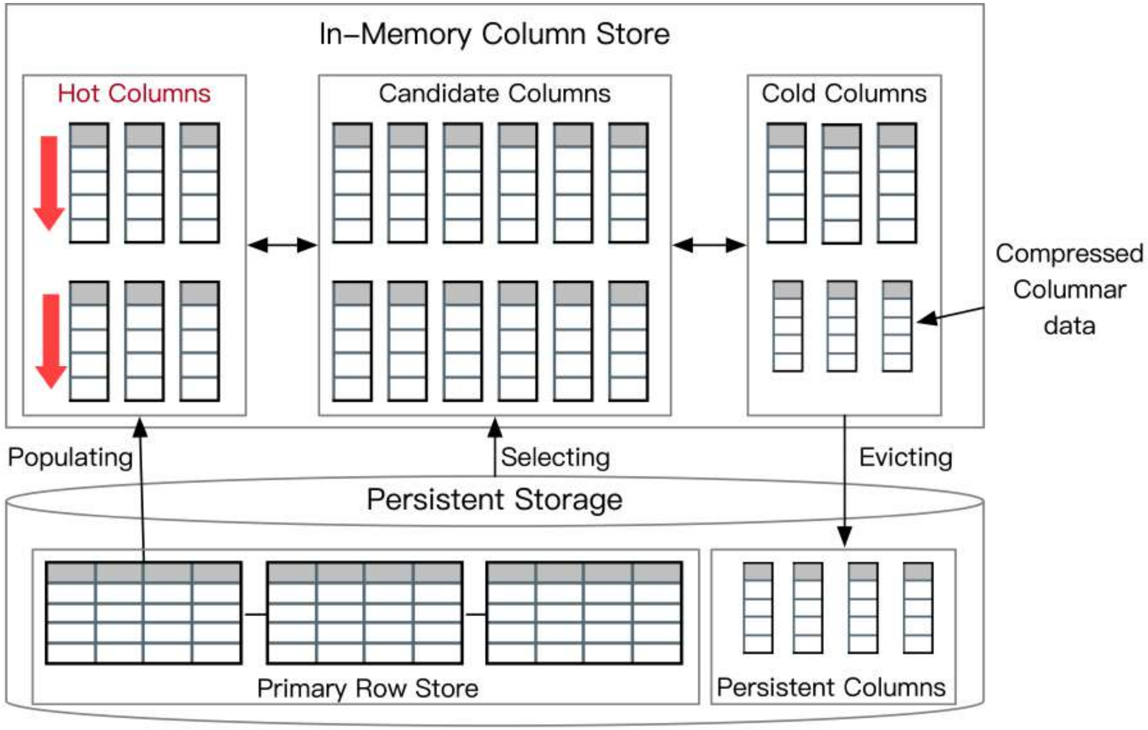
Query Optimization in HTAP databases

1. **In-Memory Column Selection**
   - **Problem : selecting which columns into the memory from the row store based on history statistics and a memory budget.**
   - **(1) Heatmap and (2) Integer programming**



Query Optimization in HTAP databases

1. **In-Memory Column Selection**
   – **Heatmap, e.g., Oracle**
   – **Basic idea: manage the columns based on the access frequencies**

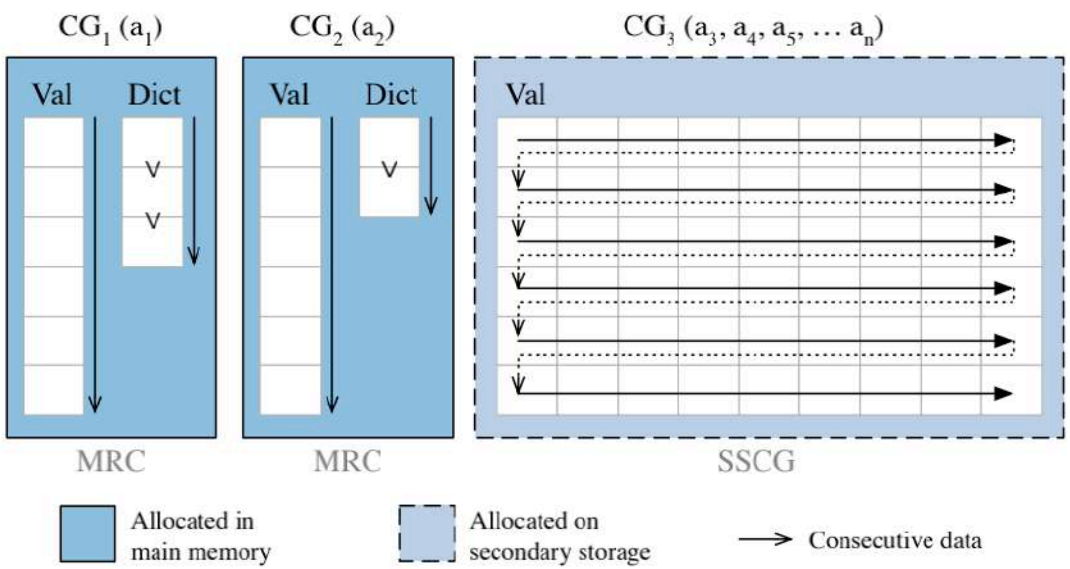


Oracle 21c. Automating Management of In-Memory Objects., 2021.

Query Optimization in HTAP databases

1. In-Memory Column Selection

- Integer programming for 0/1 Knapsack problem, e.g., Hyrise
- Basic idea: cost-based optimization problem



$$F(\vec{x}) := \sum_{j=1,\ldots,Q} b_j \cdot f_j(\vec{x})$$

$$\underset{x_i \in \{0,1\}, i=1,\ldots,N}{\text{minimize}} \quad F(\vec{x})$$

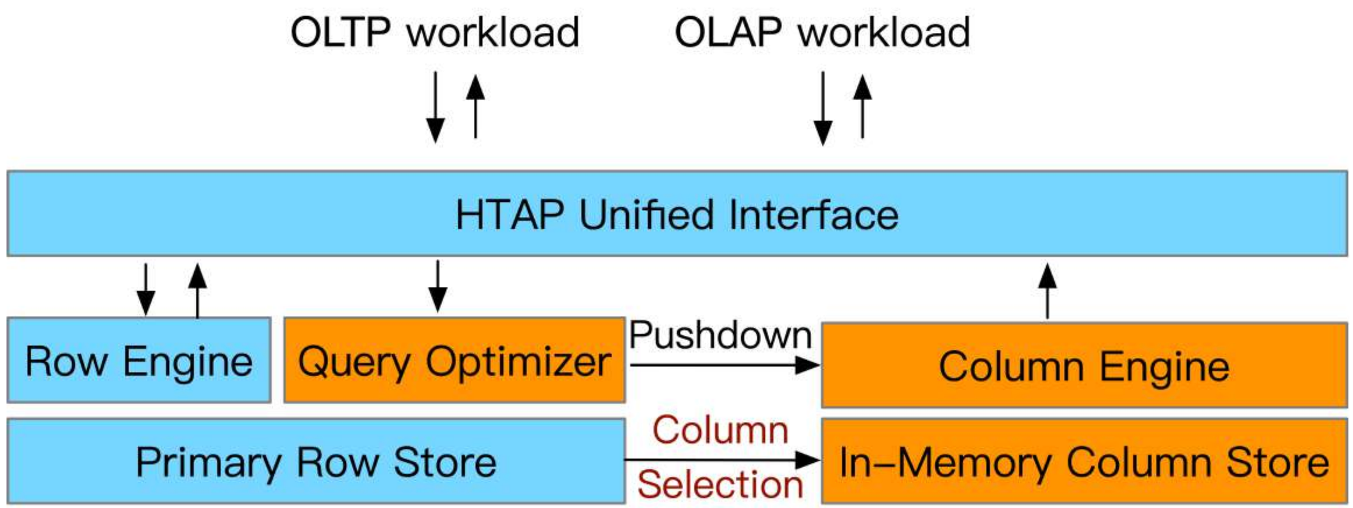
$$\text{subject to} \quad M(\vec{x}) \leq A$$

Boissier, Martin, et al. "Hybrid data layouts for tiered HTAP databases with pareto-optimal data placements." *ICDE*, 2018.

Query Optimization in HTAP databases

2. Hybrid Row and Column Scan

- Leverage hybrid row/column scan to accelerate a query
- Rule-based Plan Selection
- Cost-based Plan Selection

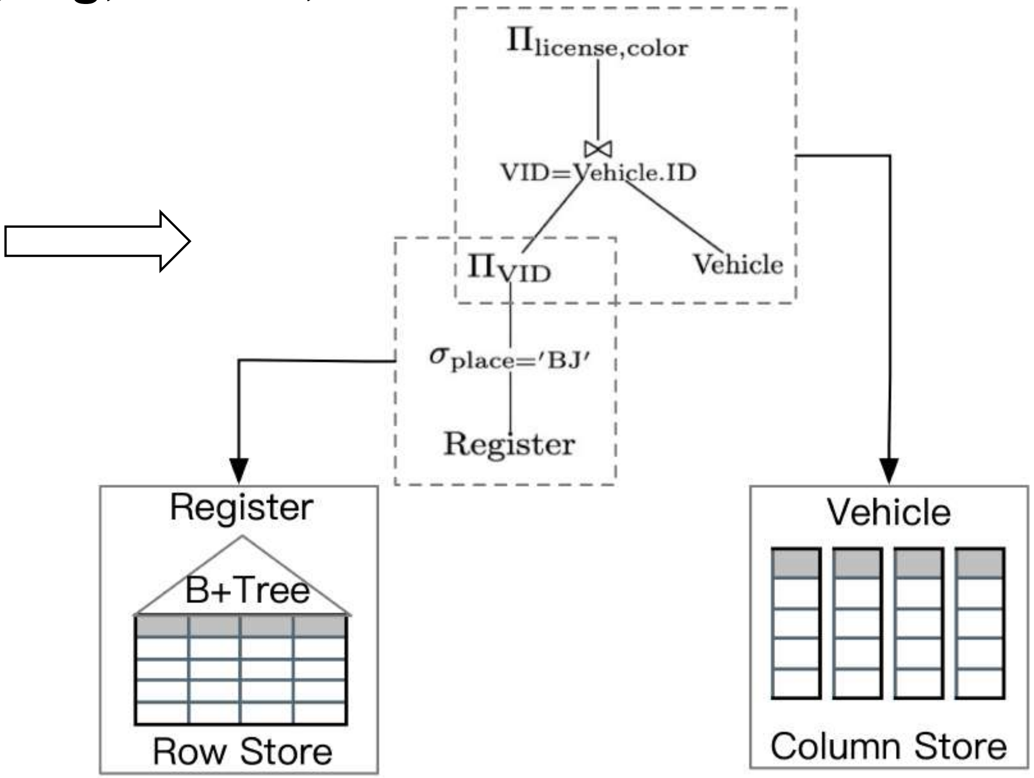


Query Optimization in HTAP databases

2. Hybrid Row and Column Scan

– **Rule-based optimization (RBO): Column Scan first, otherwise Row Scan with B+ tree search, e.g, Oracle, SQL server**

SELECT license, color
FROM Register R, Vehicle V
WHERE R.VID=V.ID and R.place='BJ'



Query Optimization in HTAP databases

2. Hybrid Row and Column Scan

– **Cost-based Optimization (CBO), e.g., TiDB**

– **Compare the cost of row/index scan with the cost of columnar scan**

SELECT T.*, S.a
FROM T, S
WHERE T.b=S.b and
1<T.a<100

$$C_{optimal} = \min(C_{column}, C_{row}, C_{index}) \quad (1)$$

$$C_{column} = \sum_{i=1}^{n} (\bar{S}_c \cdot \check{S}_i \cdot C_{scan\_column} + \check{S}_i \cdot C_{seek\_column}) \quad (2)$$

$$C_{row} = \bar{S}_r \cdot \check{S}_r \cdot C_{scan\_row} + \check{S}_r \cdot C_{seek\_row} \quad (3)$$

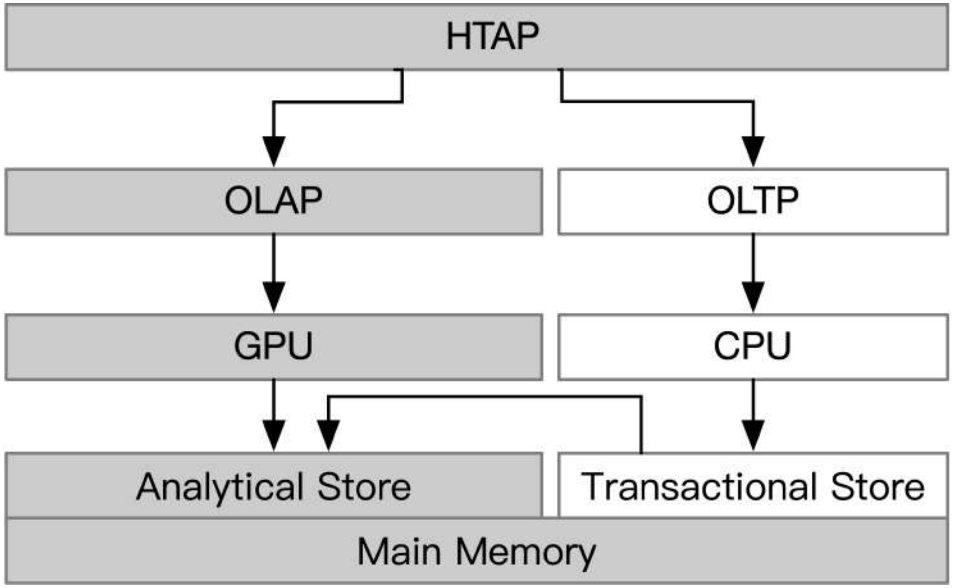
$$C_{index} = \bar{S} \cdot \check{S} \cdot C_{scan\_index} + \check{S} \cdot C_{seek\_index} + C_{double\_read} \quad (4)$$

Huang, Dongxu, et al. "TiDB: a Raft-based HTAP database." *PVLDB* 13(12), 2020: 3072-3084.

Query Optimization in HTAP databases

3. CPU/GPU Acceleration for HTAP

- CPU/GPU processors for HTAP, e.g., RateupDB, Caldera
- Task-parallel nature of CPUs for handling OLTP
- Data-parallel nature of GPUs for handling OLAP



Query Optimization in HTAP databases

Optimization Type	Databases	Techniques	Pros	Cons
In-Memory Column Selection	Oracle	HeatMap	High Efficiency	Oscillation
	Hyrise	Integer Programming	High Utility	Independent Assumption
Hybrid Row/Column Scan	Oracle	Rule-based Plan Selection	High Efficiency	Local Search
	TiDB	Cost-based Plan Selection	Global Search	Independent Assumption
CPU/GPU Acceleration	Caldera	Copy-on-Write	High Freshness	Low Throughput
	RateupDB	Isolated Dual Store with Data Sync	High Throughput	Low Freshness

Resource Scheduling

1. Workload-driven Scheduling for HTAP

- Scheduling based on the executed performance of HTAP workloads
- E.g., assign more threads and memory for OLTP or OLTP.
- SAP HANA

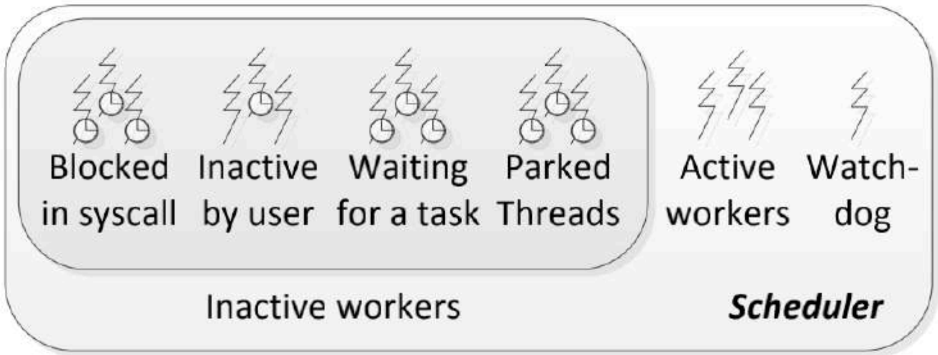
2. Freshness-driven Scheduling for HTAP

- Switches the modes on workload execution
- Trade-off between data freshness and workload isolation
- Resource and Data Exchange (RDE) [Raza, Aunn, et al SIGMOD 20]

Resource Scheduling

1. Workload-driven Scheduling for HTAP

- ❑ Assign more threads to OLTP or OLAP
- ❑ Isolate the workload execution and assign more cache for OLAP



Psaroudakis, Iraklis, et al. "Task scheduling for highly concurrent analytical and transactional main-memory workloads." *In ADMS*, 2013.

		LLC partition size for OLTP in MBs				
		1.75	8.75	17.5	31.5	33.25
Projection	OLTP	0.68	0.73	0.77	0.83	0.86
	OLAP	1.00	1.00	1.00	0.98	0.47
Join	OLTP	0.77	0.81	0.83	0.86	0.84
	OLAP	1.00	0.99	0.98	0.90	0.68

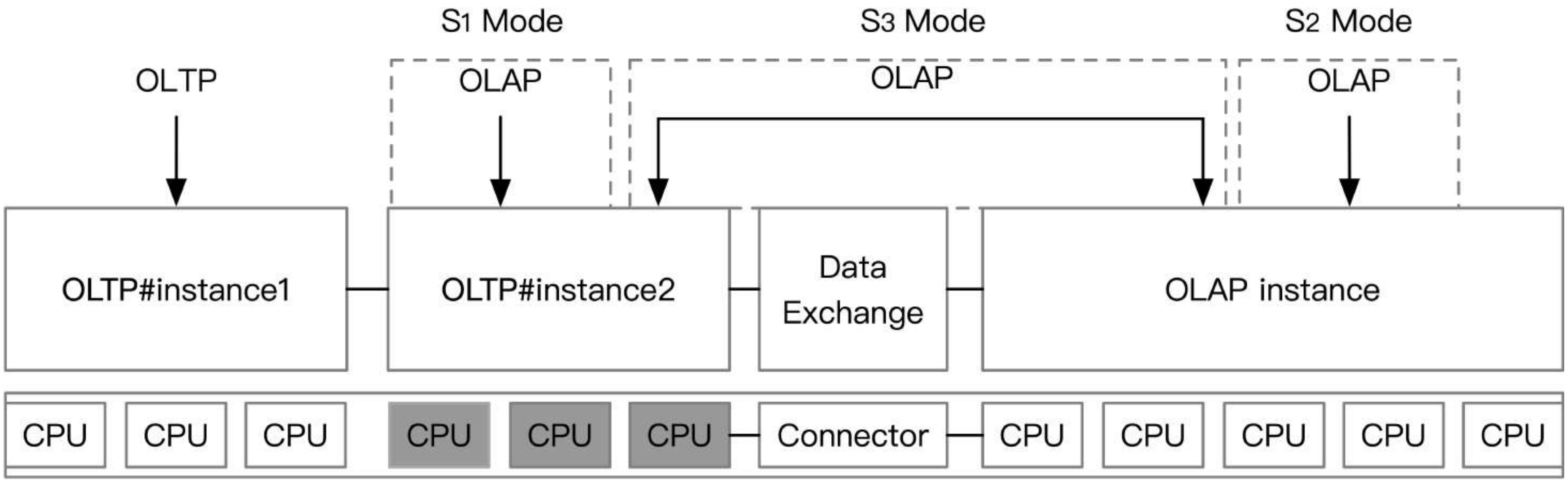
Sirin, Utku, Sandhya Dwarkadas, and Anastasia Ailamaki. "Performance Characterization of HTAP Workloads." *In ICDE*, 2021.

Resource Scheduling

2. Freshness-driven Scheduling for HTAP

– Switches the execution modes {S1, S2, S3}

– Default S2; When freshness < threshold, jump to S1 or S3



Raza, Aunn, et al. "Adaptive HTAP through elastic resource scheduling." *In SIGMOD*, 2020.

Resource Scheduling in HTAP databases

Scheduling Type	Databases /Prototype	Techniques	Pros	Cons
Workload-Driven Scheduling	SAP HANA, Siper	Rule-based Scheduling	High Utility	Low Freshness
Freshness-Driven Scheduling	RDE	Threshold-based Execution Switching	High Freshness	Oscillation

A Summary of HTAP Key Techniques

Task Type	Key Techniques	HTAP Databases	Pros	Cons
Transaction Processing	Standalone TP with In-Memory Delta Update	Oracle, SQL Server, SAP HANA	High Efficiency	Low Scalability
	Distributed TP with Log Reply	F1 Lightning, TiDB, SingleStore	High Scalability	Low Efficiency
Analytical Processing	Standalone columnar scan with in-memory delta traversing	Oracle, SQL Server, SAP HANA	High Freshness	Large Memory Size
	Distributed columnar scan with log file scanning	F1 Lightning, TiDB	High Scalability	Low Efficiency

A Summary of HTAP Key Techniques (cont.)

Task Type	Key Techniques	HTAP Databases	Pros	Cons
Data Synchronization	In-Memory Delta Merge	Oracle, SQL Server, SAP HANA	High Efficiency	Low Scalability
	Log-based Delta Merge	F1 Lightning, TiDB	High Scalability	High Merge Cost
Query Optimization	In-Memory Column Selection	Oracle, MySQL Heatwave	High Memory Utility	Low AP Throughput
	Hybrid Row/Column Scan	TiDB, Oracle, SQL Server	High AP Throughput	Large Search Space
	CPU/GPU Acceleration for HTAP	RateupDB, Caldera	High AP Throughput	Low TP Throughput
Resource Scheduling	Freshness-Driven Scheduling	SAP HANA, Siper	High Freshness	Low Throughput
	Workload-Driven Scheduling	RDE	High Throughput	Low Freshness

Other Relevant HTAP Techniques

Other Relevant HTAP Techniques

❑ Multi-Versioned Indexes for HTAP

– **Parallel Binary Tree (P-Tree), Multi-Version Partitioned B-Tree (MV-PBT)**

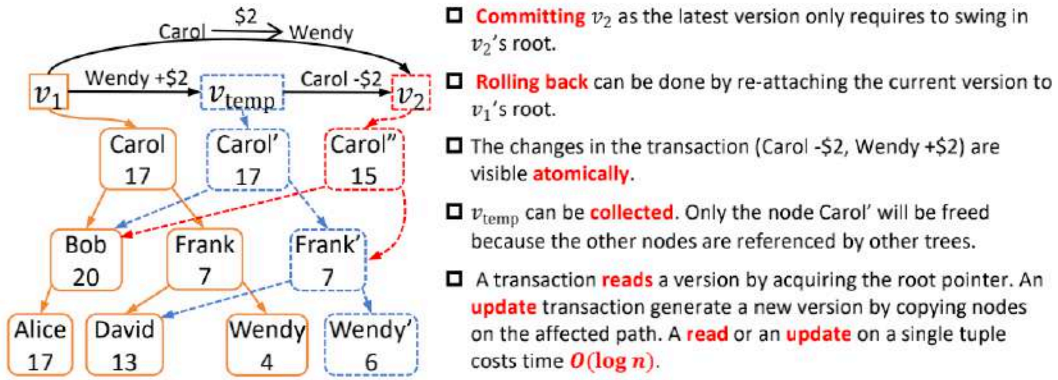


Figure 1: P-Trees Multi-Versioning Overview – An example of using P-Trees to support a bank balance DBMS. The original state is v_1. A transaction transfers \$2 from Carol to Wendy.

Sun, Yihan, et al. "On supporting efficient snapshot isolation for hybrid workloads with multi-versioned indexes." *PVLDB* 13(2), 2019.

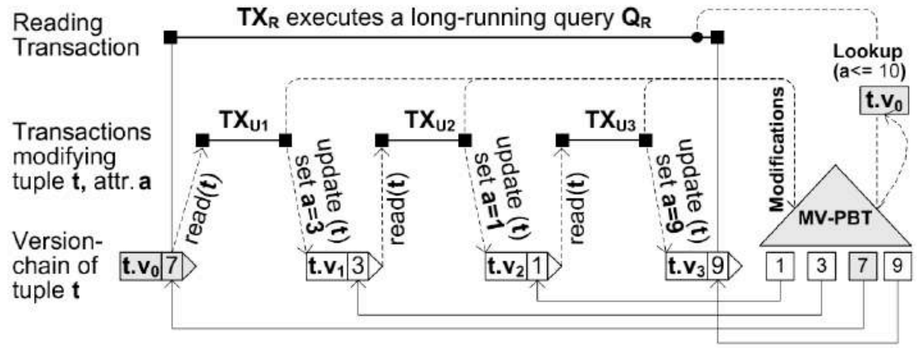


Figure 1: HTAP and Version-Chain Lengths: $TX_{U1} \ldots TX_{U3}$ create new versions of tuple *t*, which are indexed. The index scan of TX_R returns only the index entries ($t.v_0$) visible to TX_R filtering the invisible ones ($t.v_1 \ldots t.v_3$), matching the search predicate.

Riegger, Christian, et al. "MV-PBT: multi-version indexing for large datasets and HTAP workloads." *In EDBT*, 2020.

Other Relevant HTAP Techniques

- **Adaptive Data Organization for HTAP**
  - H2O [Alagiannis et al, SIGMOD 2014], Casper [Athanassoulis et al, VLDB 2019]
  - Peloton [Arulraj et al, SIGMOD 2016], Proteus [Abebe et al, SIGMOD 2022]

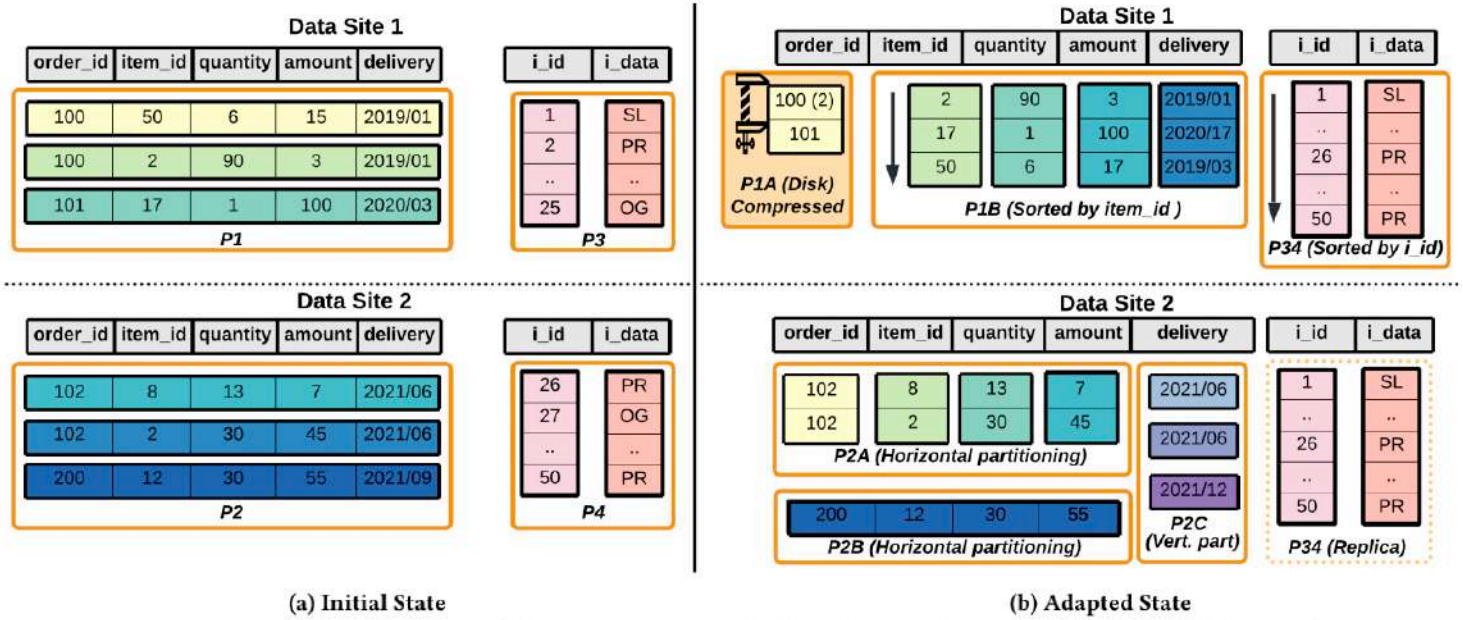


Figure 4: An initial storage layout of data in Proteus, and adapted state after a series of storage layout changes.

Abebe, Michael, Horatiu Lazu, and Khuzaima Daudjee. *Proteus: Autonomous Adaptive Storage for Mixed Workloads*. In SIGMOD, 2022.

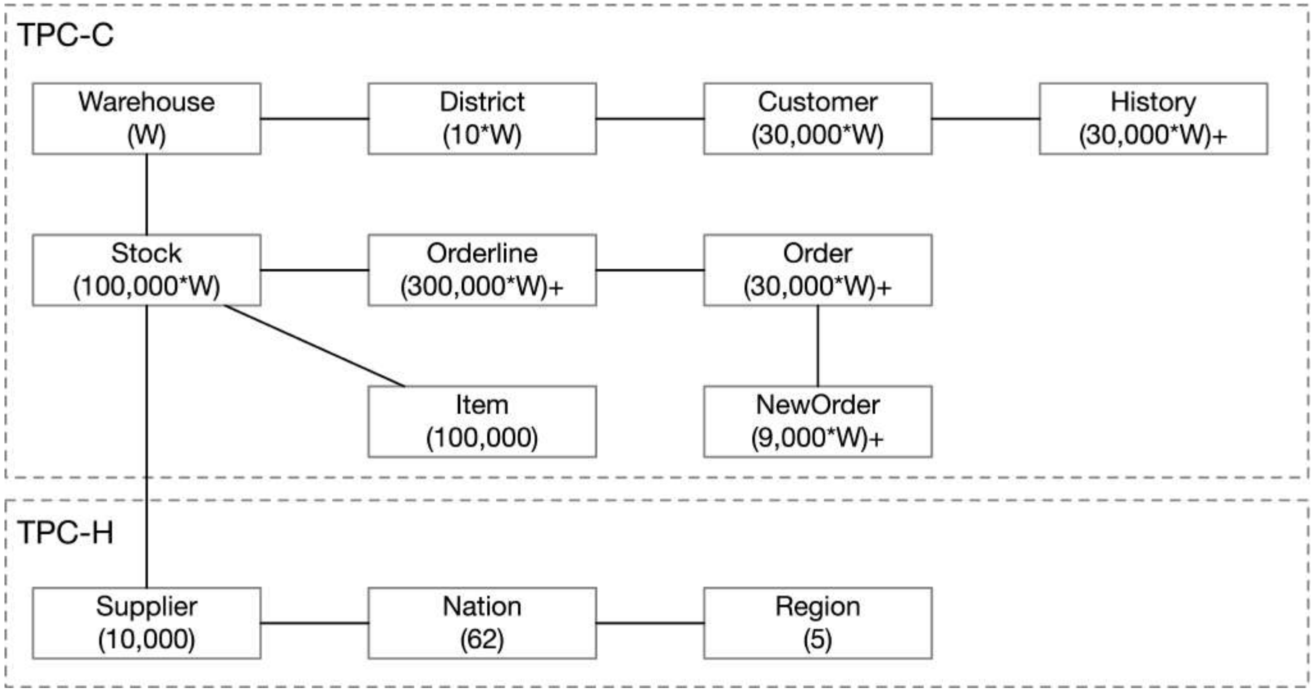
HTAP Benchmarks

Overview of HTAP Benchmarks

- CH-Benchmark
  - TPC-C + TPC-H
  - Controllable Hybrid OLTP/OLAP Workload Execution
  - Unified Metric
- HTAPBench
  - TPC-C + TPC-H
  - Fixed OLTP Metric with Controllable OLAP Workload Execution
  - Time Window
  - Unified Metric

CH-Benchmark

- Unified Schema of TPC-C and TPC-H
- 12 tables by merging TPC-C's 9 tables with TPC-H's 8 tables
- Scaling TPC-H by the same factors of TPC-C



CH-Benchmark

- **TPC-C transactions**
  - New Order, Payment, Order-Status, Delivery, and Stock-Level.
- **Modified TPC-H 22 queries**
  - Modified table name and join key,
  - Less arithmetic operations
- **Removed refresh functions**

```
SELECT n_name, SUM(ol_amount) AS revenue
FROM customer, "order", orderline, stock, supplier,
  nation, region
WHERE c_id=o_c_id AND c_w_id=o_w_id AND c_d_id=o_d_id
  AND ol_o_id=o_id AND ol_w_id=o_w_id
  AND ol_d_id=o_d_id
  AND ol_w_id=s_w_id AND ol_i_id=s_i_id
  AND mod((s_w_id * s_i_id),10000)=su_suppkey
  AND ascii(SUBSTRING(c_state, 1, 1))=su_nationkey
  AND su_nationkey=n_nationkey
  AND n_regionkey=r_regionkey
  AND r_name='[REGION]' AND o_entry_d>='[DATE]'
GROUP BY n_name ORDER BY revenue DESC
```

Listing 1: CH-benCHmark query 5

```
SELECT n_name, SUM(l_extendedprice * (1 - l_discount)
  ) AS revenue
FROM customer, orders, lineitem, supplier, nation,
  region
WHERE c_custkey = o_custkey
  AND l_orderkey = o_orderkey
  AND l_suppkey = s_suppkey
  AND c_nationkey = s_nationkey
  AND s_nationkey = n_nationkey
  AND n_regionkey = r_regionkey
  AND r_name = '[REGION]'
  AND o_orderdate >= DATE '[DATE]'
  AND o_orderdate < DATE '[DATE]' + INTERVAL '1' YEAR
GROUP BY n_name ORDER BY revenue DESC
```

Listing 2: TPC-H Query 5

CH-Benchmark

- **Execution Rules**
  - OLTP/OLAP only or Hybrid Execution (OLTP+OLAP)
  - The number of parallel OLTP and OLAP streams
- **Metrics**
  - OLAP-oriented metric

$$M(OLAP) = \boxed{\frac{\text{tpmC}}{\text{QphH}}} @ \boxed{\text{QphH}}$$

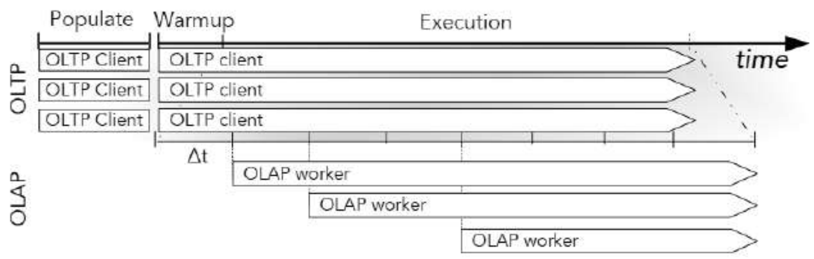
  - OLTP-oriented metric

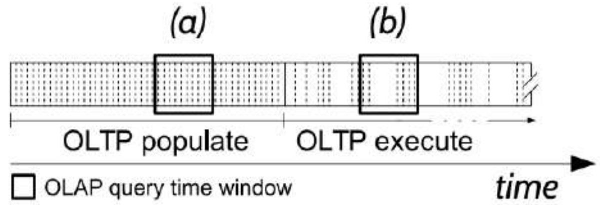
$$M(\text{OLTP}) = \frac{\text{tpmC}}{\text{QphH}} @\text{tpmC}$$

- **For example, Isolated execution is 5.7@5084 tpmC, hybrid execution is 6.5@5188 tpmC, indicating the OLTP throughput increased with hybrid execution.**

HTAPBench

- **Schema and Workload**
  - **The same as CH-Benchmark (TPC-C+ TPC-H)**
- **Execution Rules**
  - **Fixed target tpmC with controllable OLAP workers**
  - **Time window for querying newly-inserted data**
- **Metric**
  - **Under a certain TP throughput, the AP throughput per hour per worker**





$$\mathrm{QpHpW} = \boxed{\frac{\mathrm{QphH}}{\#\mathrm{OLAPworker}}} @ \boxed{\mathrm{tpmC}}$$

Comparison of HTAP End-to-End Benchmarks

HTAP Benchmark	Data and Workload	Techniques	Pros	Cons
CH-Benchmark	TPC-C+TPC-H	Controllable Hybrid Execution, Comparable Unified Metric	Flexible Execution	Low Data Freshness
HTAPBench	TPC-C+TPC-H	Fixed OLTP Metric with Controllable OLAP Execution, Dynamic Time Window	High Data Freshness	Fixed OLTP metric

Other HTAP Benchmarks

- **Swarm64 HTAP benchmark**
  - Combine CH-benchmark + HTAPBench
  - Hybrid Execution from CH-benchmark
  - Dynamic Time Window from HTAPBench
- **Micro-Benchmarks**
  - ADAPT Benchmark [Arulraj et al, SIGMOD 2016]
  - HAP Benchmark [Athanassoulis et al, VLDB 2019]

Q_1: **INSERT INTO** R **VALUES** $(a_0, a_1, \ldots, a_p)$
Q_2: **SELECT** $a_1, a_2, \ldots, a_k$ **FROM** R **WHERE** $a_0 < \delta$
Q_3: **SELECT** MAX(a_1),...,MAX(a_k) **FROM** R **WHERE** $a_0 < \delta$
Q_4: **SELECT** $a_1 + a_2 + \ldots + a_k$ **FROM** R **WHERE** $a_0 < \delta$
Q_5: **SELECT** X.a_1,...,X.a_k, Y.a_1,...,Y.a_k
FROM R **AS** X, R **AS** Y **WHERE** X.a_i < Y.a_j

Workload of ADAPT Benchmark

Q_1: **SELECT** $a_1, a_2, \ldots, a_k$ **FROM** R **WHERE** $a_0 = v$
Q_2: **SELECT** $count(*)$ **FROM** R **WHERE** $a_0 \in [v_s, v_e)$
Q_3: **SELECT** $a_1 + a_2 + \cdots + a_k$ **FROM** R **WHERE** $a_0 \in [v_s, v_e)$
Q_4: **INSERT INTO** R **VALUES** $(a_0, a_1, a_2, \ldots, a_p)$
Q_5: **DELETE FROM** R **WHERE** $a_0 = v$
Q_6: **UPDATE** R **SET** $a_0 = v_{new}$ **WHERE** $a_0 = v$

Workload of HAP Benchmark

Open Problems and Opportunities

Open Problem #1: Data Organization for HTAP Databases

- Primary Row Store for TP, Replicated Column Store for AP
  - Column Selection Problem (Memory-Bounded, NP-Hard)
  - Column Compression Problem (Memory-Bounded, LZ4, Dictionary, Bit-Packing)

- Adaptive Storage for HTAP workloads
  - – Fine-grained horizontal and vertical data partitioning for HTAP
  - – Need to be further justified in practice due to the high complexity

Open Problem #2: Data Synchronization for HTAP

- **Data Synchronization Problem**
  - **Synchronization Order (Which data to synchronize)**
  - **Time Interval Decision (When to synchronize the data)**
  - **New Data Structures and Merging Strategies (How to traverse and merge the delta)**

- **Possible solutions: Learned cost model; Prediction model for workload trend; B-tree and LSM-tree Indexing**

Open Problem #3: Optimization for Row & Column Scan

- **Pushdown analytical operators to Columnar Scan**
  - **Compare the cost between row scan and columnar scan in the operator level**
  - **Need to design cost functions without independent and uniform assumption**

- **Possible solutions: Reinforcement learning for exploring the plan space; Learned cost estimator for correlated and skewed distribution**

Open Problem #4: HTAP Resource Scheduling

- **Adaptive HTAP Resource Scheduling**
  - **Consider both system performance and data freshness**
  - **Need to adapt to new workload access patterns**

- **Possible solutions: Pretrained cost model; lightweight model for new workload pattern; Serverless computing for HTAP workloads.**

Open Problem #5: HTAP Benchmark Suite

- HTAP Benchmark Suite
  - Controllable and extensible HTAP benchmark
  - Need to add more workload patterns and datasets
  - Need to add more micro-benchmark for HTAP

- Possible solutions: Unified HTAP benchmark suite, e.g., OLTPBench; Add analytical operations into TPC-C, replacing TPC-H with JCC-H and TPC-DS; new micro-benchmarks for data organization, data synchronization, query optimization, resource scheduling, etc.

Thanks! Q & A

Main Rererences (1/3)

- M. Pezzini, D. Feinberg, N. Rayner, and R. Edjlali. Hybrid Transaction/Analytical Processing Will Foster Opportunities For Dramatic Business Innovation. Gartner, pages 4–20, 2014.
- D. Feinberg. Setting the Record Straight – HTAP OPDBMS, 2018.
- F. Özcan, Y. Tian, and P. Tözün. Hybrid Transactional/Analytical Processing: A Survey. In SIGMOD, pages 1771–1775, 2017.
- 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.
- M. Grund, J. Krüger, H. Plattner, A. Zeier, P. Cudre-Mauroux, S. Madden. Hyrise: a main memory hybrid storage engine. Proceedings of the VLDB Endowment, 4(2):105-116, 2010.
- A. Kemper, T. Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In ICDE, pages 195-206, 2011.
- T. Neumann, T. Mühlbauer, and A. Kemper. Fast Serializable Multi-Version Concurrency Control for Main-Memory Database Systems. In SIGMOD, pages 677–689, 2015.

Main Rererences (2/3)

- 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.
- 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.
- T. Lahiri, S. Chavan, M. Colgan, D. Das, A. Ganesh, M. Gleeson, S. Hase, et al. Oracle Database In-Memory: A Dual Format In-Memory Database. In ICDE, pages 1253–1258. IEEE, 2015.
- 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.
- 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.
- Yang, J., Rae, I., Xu, J., Shute, J., Yuan, Z., Lau, K., ... & Cieslewicz, J. (2020). F1 Lightning: HTAP as a Service. Proceedings of the VLDB Endowment, 13(12), 3313-3325.
- PolarDB. PolarDB HTAP Real-Time Data Analysis Technology Decryption, 2021.

Main Rererences (3/3)

- R. Appuswamy, M. Karpathiotakis, D. Porobic, and A. Ailamaki. The Case For Heterogeneous HTAP. In CIDR, 2017.
- 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.
- MySQL Heatwave. Real-time Analytics for MySQL Database Service, 2021.
- 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 TPCTC, pages 97–112, 2014.
- U. Sirin, S. Dwarkadas, and A. Ailamaki. Performance Characterization of HTAP Workloads. In ICDE, pages 1829–1834. IEEE, 2021.
- A. Raza, P. Chrysogelos, A. C. Anadiotis, and A. Ailamaki. Adaptive HTAP Through Elastic Resource Scheduling. In SIGMOD, pages 2043–2054, 2020.
- 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, pages 293– 304, 2017.
- 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 *DBTest*, pages 1–6, 2011.