VLDB2024
GUANGZHOU

HyBench : A Benchmark for HTAP Databases

Chao Zhang, Guoliang Li*, Tao Lv

Tsinghua University

中国软件评测中心

Table of Content

Table of Content

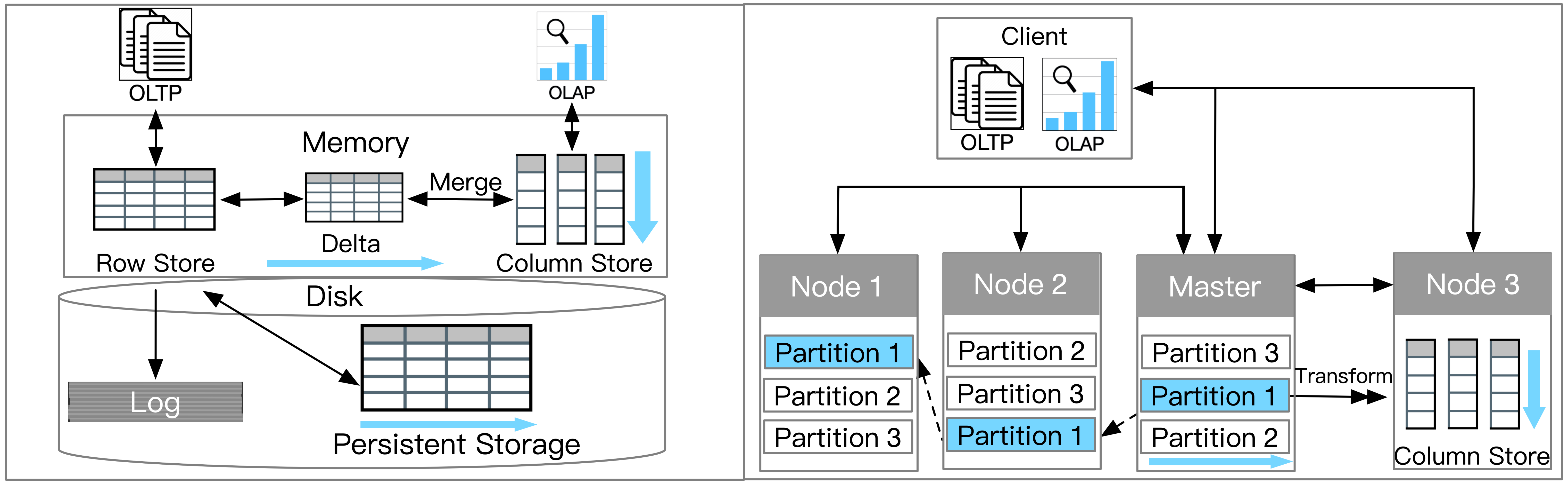
Four Main HTAP Architectures

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

(b) Distributed Row Store + Column Store Replica

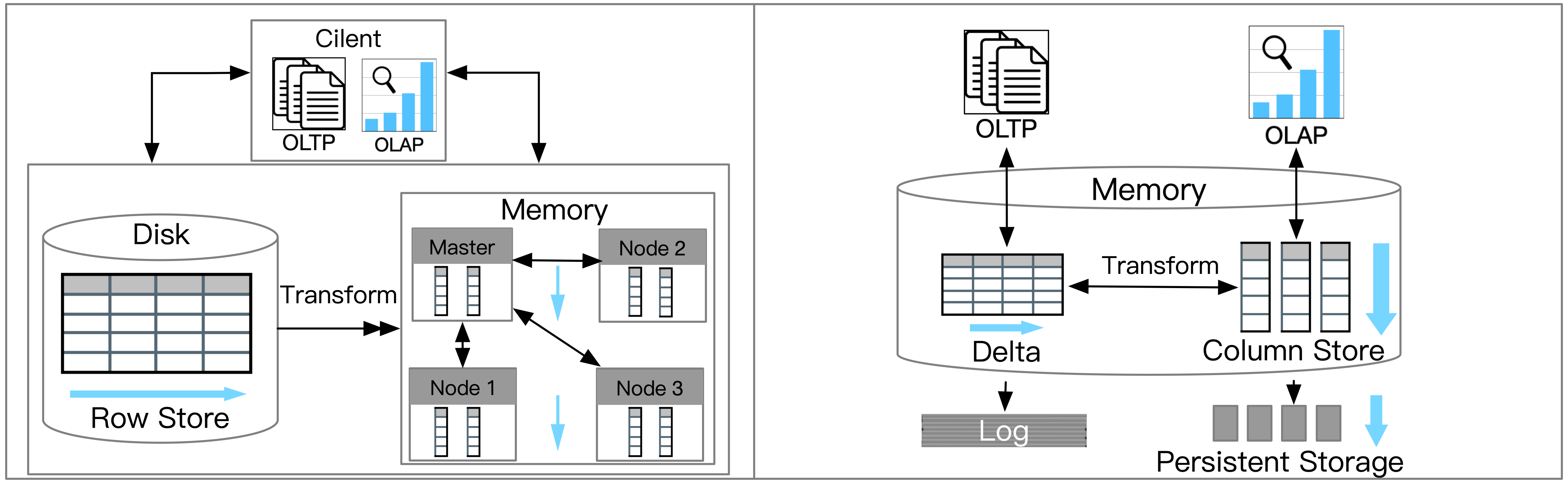
(c) Disk Row Store + Distributed Column Store

(d) Primary Column Store + Delta Row Store



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

(b) Distributed Row Store+Column Store Replica



It is crucial to have a benchmark that can evaluate the pros and cons of different HTAP architectures

Existing Approaches

Benchmark	Schema/Workload	Key Techniques	Pros	Cons
CH-Benchmark	TPC-C+TPC-H	Unified Schema with Hybrid Execution	Usability	No freshness metric
HTAPBench	TPC-C+TPC-H	Fixed OLTP with Controllable OLAP Execution	Fresh data access	No freshness metric
OLXPBench	TPC-C& Smallbank& TATP	Adding queries to existing OLTP Benchmarks	New HTAP workload	No freshness metric
HATtrick	TPC-C+SSB	Unified Schema with Hybrid Execution	Freshness evaluation	High overhead

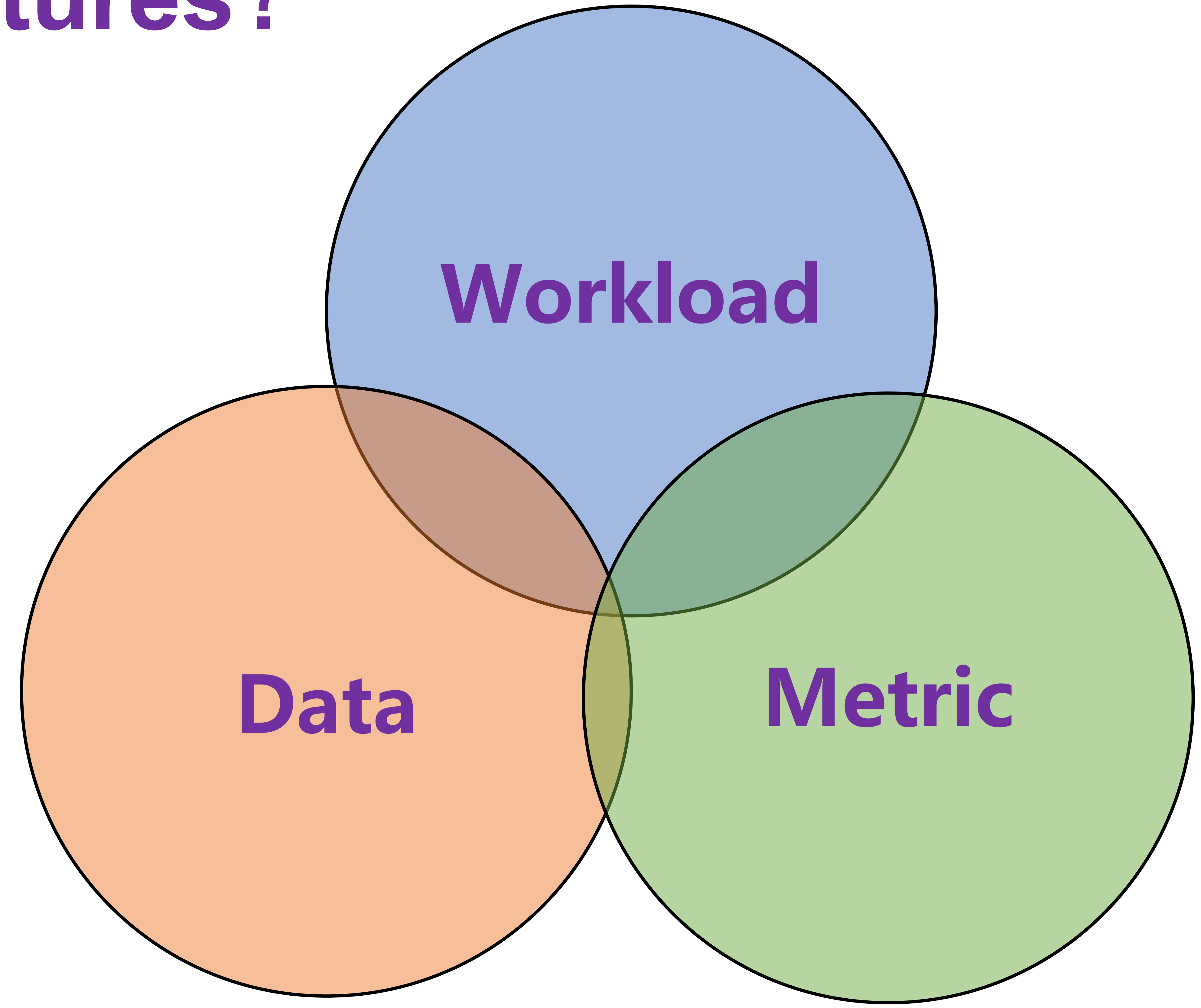
Existing Approaches fall short of tailored schema, workload, and metrics

Towards HTAP Benchmarking

Research Problem：How do we make a holistic evaluation concerning various HTAP architectures？

Research Questions：

- RQ1:How to generate realistic data？
- RQ2:How to synthesize hybrid workload?
- RQ3:How to design holistic HTAP metrics?



Challenge 1

Data

Workload

Metric

Challenge 1：How to generate realistic HTAP data due to the impedance mismatch between existing benchmarks and the realistic applications and data?

Our Solution: HTAP-Native Application with Query-Driven Generation

Challenge 2

Data → Workload → Metric

Challenge 2：How to synthesize tailored hybrid workload due to the large design space and the complexity of controlling data contention among the transactions and queries?

Our Solution: HTAP-Native Hybrid Workload with Controllable Contention

Challenge 3

Data → Workload → **Metric**

Challenge 3：How to evaluate the freshness efficiently and effectively during the throughput evaluation?

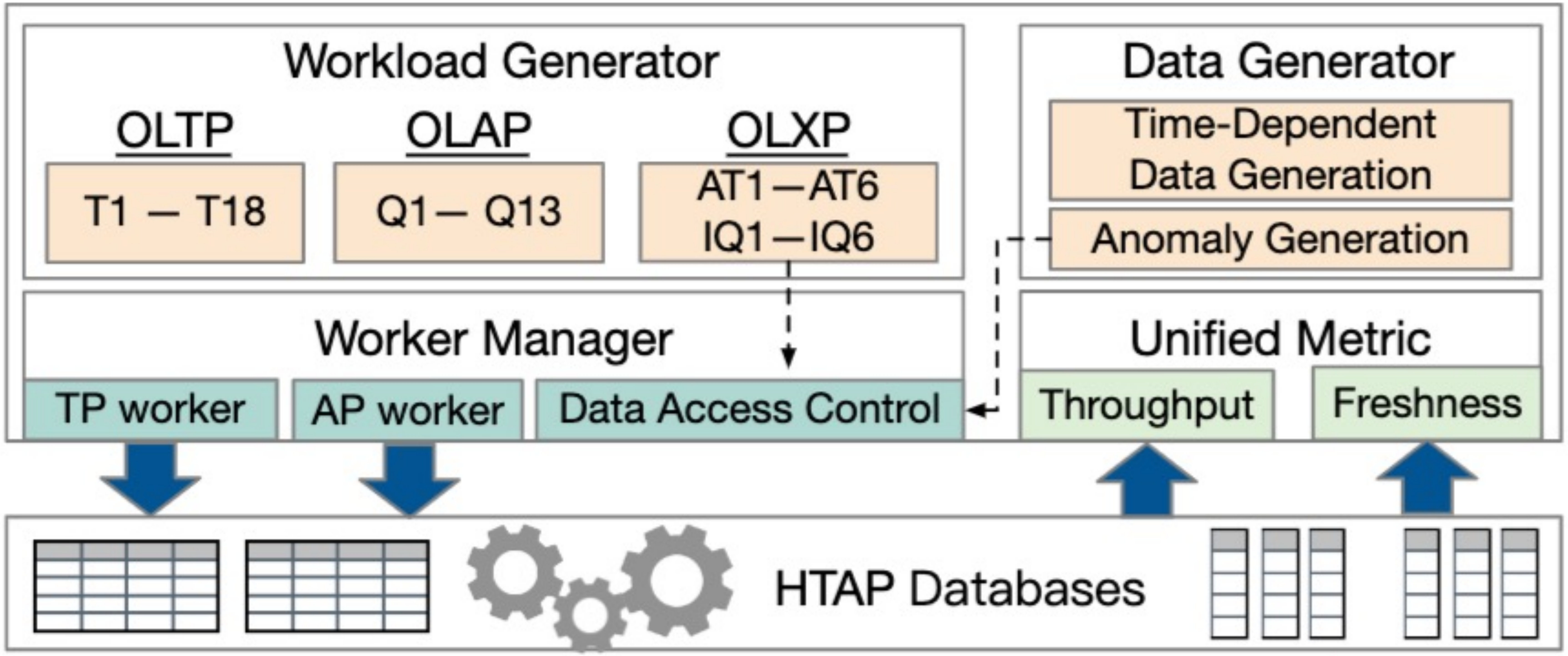
Our Solution: Query Result Driven Method on Freshness Evaluation

Table of Content

- Background and Motivation
- HyBench to the Rescue
  - Schema and Data Generation
  - Workload Design and Control
  - HTAP Metrics
- Evaluation
- Conclusion

HyBench Overview

- Data Generator: time-dependent and anomaly generation
- Workload Generator: a hybrid workload of OLTP, OLAP, and OLXP with choke-point design
- Unified Metric: An E2E Evaluation of Freshness, TPS, QPS, XPS

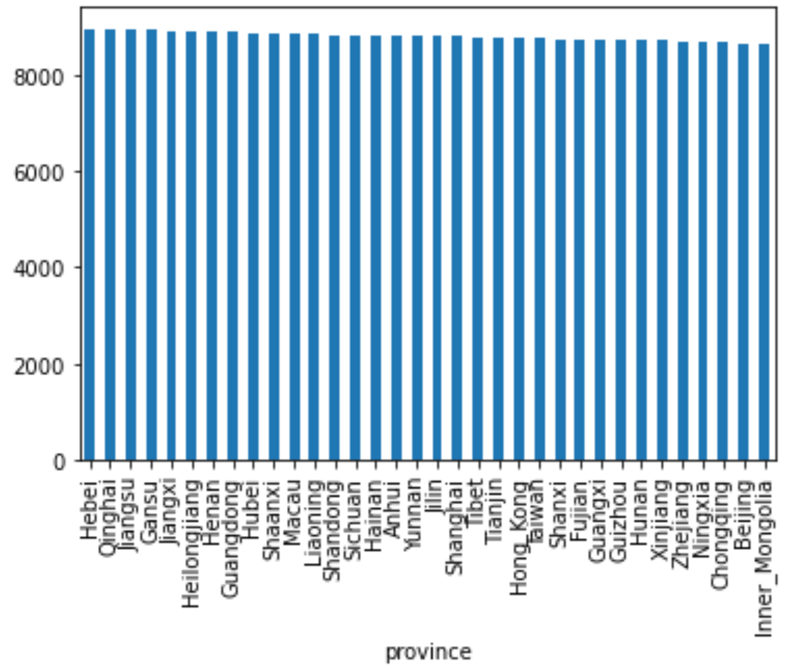


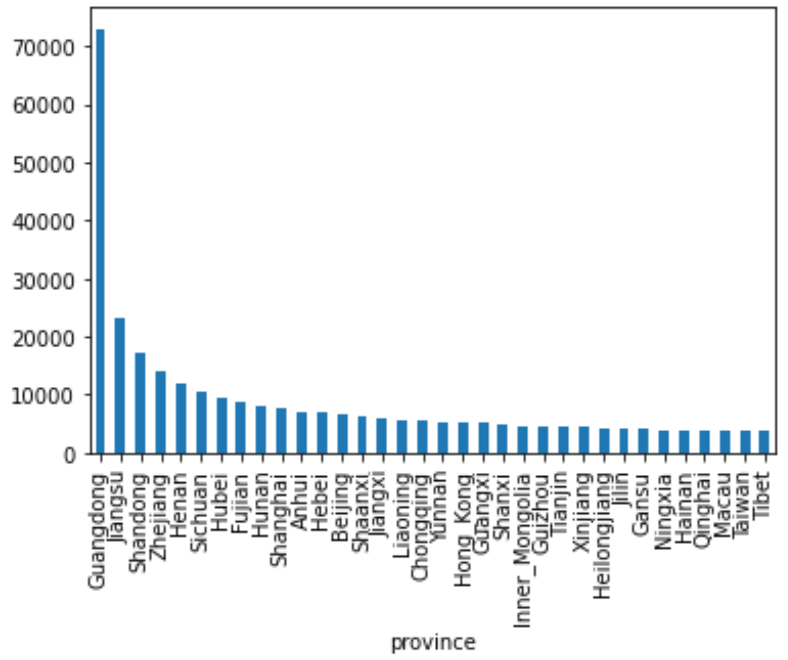
HyBench Schema

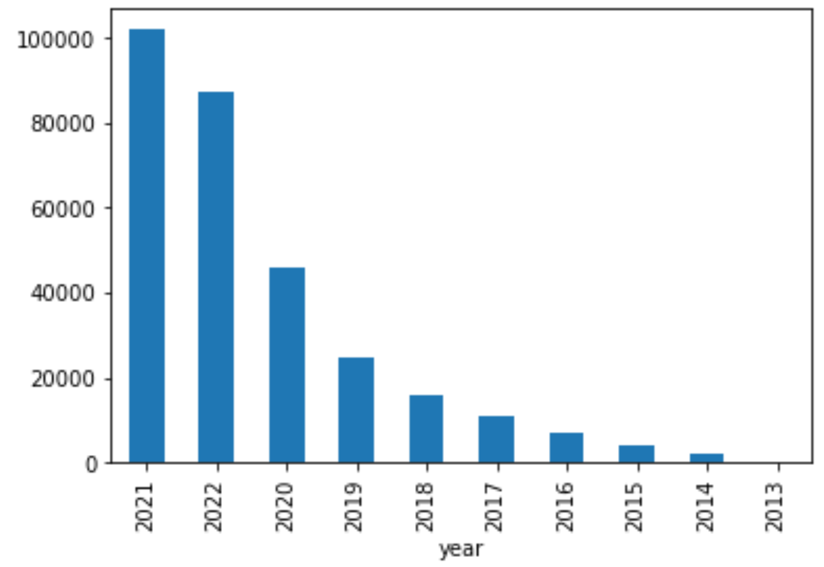
- **Applications:** On-line Finance APPs

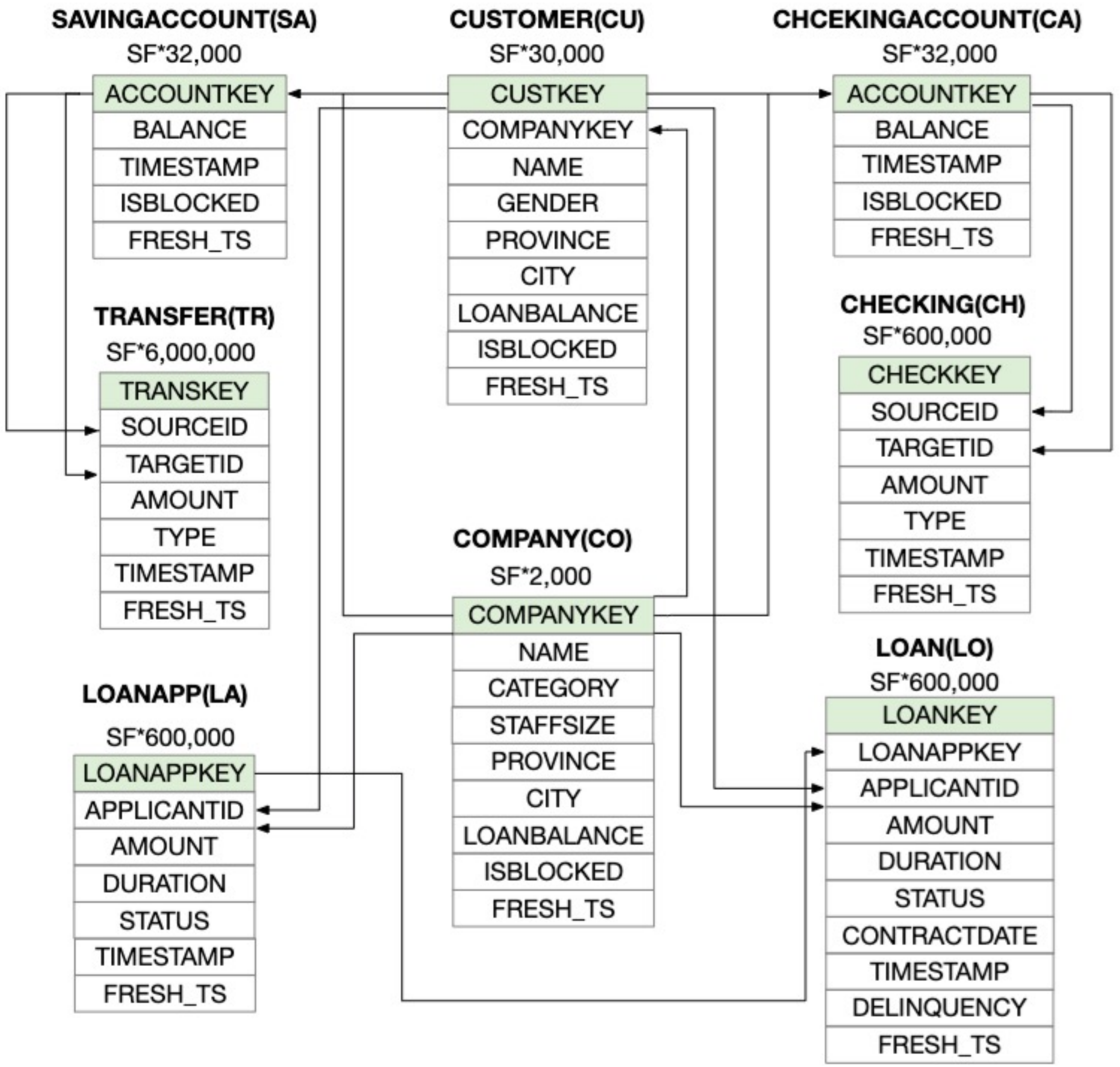
支付宝 ALIPAY WeBank 微众银行

- **Multi-Level:** Customer->Company->Vendor
- **Distributions:** Uniform and Skewed









HyBench Data Generation

- **Phase 1:** it generates the base data and anomaly data of CU, CO, SA, CA.
- **Phase 2:** it generates the base data and anomaly data of TR and CH; we use a query-driven method to produce the anomalies.
- **Phase 3:** it generates the base data and anomaly data of LA and LO; we can control the number of delayed loans with probability p_2

Algorithm 1: Time-Dependent Data Generation

Input: Scale Factor: SF, Dates: d_1, d_2, d_3, d_4, Probabilities: p_1, p_2
Output: A HyBench Dataset D and Anomaly Parameters $\hat{A}$.

$\mathbb{N} = \mathcal{S}(SF)$; `// Get all the scaling models`
`// Phase one: generate the base data`
$CU, CO, SA, CA = \text{DataGen}(seed, d_1, d_2, \mathbb{N}_{CU,CO,SA,CA})$;
$\hat{CU}, \hat{CO}, \hat{SA}_1, \hat{CA}_1 = \text{AnomalyGen}(p_1, CU, CO, SA, CA)$;
`// Phase two: generate the payment transactions`
$TR, CH = \text{DataGen}(seed, d_2, d_3, \mathbb{N}_{TR,CH})$;
$\hat{TR}, \hat{CH}, \hat{SA}_2, \hat{CA}_2 = \text{AnomalyGen}(TR, CH, IQ, \hat{SA}_1, \hat{CA}_1)$;
`// Phase three: generate the loan transactions`
$LA, LO = \text{DataGen}(seed, d_3, d_4, \mathbb{N}_{LA,LO})$;
$Duration = [30, 60, 90, 180, 365]$;
$CuratedDate = \text{DateCurate}(p_2, \text{Duration}, d_4)$;
$\hat{LA}, \hat{LO} = \text{AnomalyGen}(seed, CuratedDate, d_4, Duration)$;
$\hat{A} = Reservoir\_sampling(\hat{SA}_1, \hat{SA}_2, \hat{CA}_1, \hat{CA}_2, \hat{TR}, \hat{CH}, \hat{LA}, \hat{LO})$;
return $D = \{CU, CO, SA, CA, TR, CH, LA, LO\}, \hat{A}$;

Workload Design

Category	Description	Choke Points
OLTP	18 Operational Transactions	ACID Test, Row Storage, Batch Write, Hot Spot, Chain Logic, Concurrency Control for High Parallelism
OLAP	13 Analytical Queries	Window Function, Dependent Group-by, Flattening Subqueries, Large IN, Join Ordering, CTE Pushdown
OLXP	6 Analytical Transactions (AT)	Joins and Aggregations within the Transaction, Join Cycle, Left Join Optimization, Concurrency Control
	6 Interactive Queries (IQ)	Few Column Selection, Bi-directional Search, Left Join Optimization, Result Reuse, Join Ordering
	Hybrid Execution (AT & IQ)	High Data and Resource Contention, Long Chain Traversing, Skewed Data Access, Data Freshness

```
BEGIN;  # Begin the transaction
IF(Isblocked==1 or balance< @a)
ROLLBACK;  # Rollback the transaction
SELECT COUNT(*) AS CNT
FROM Transfer tr, SavingAccount sa
WHERE sa.isblocked=1
AND tr.targetId=sa.accountId
AND tr.sourceId=@targetId;
IF (CNT>0) ROLLBACK; # Risk controlling
sourceId.balance = sourceId.balance - @a
targetId.balance = targetId.balance + @a
COMMIT; # Commit the transaction
```

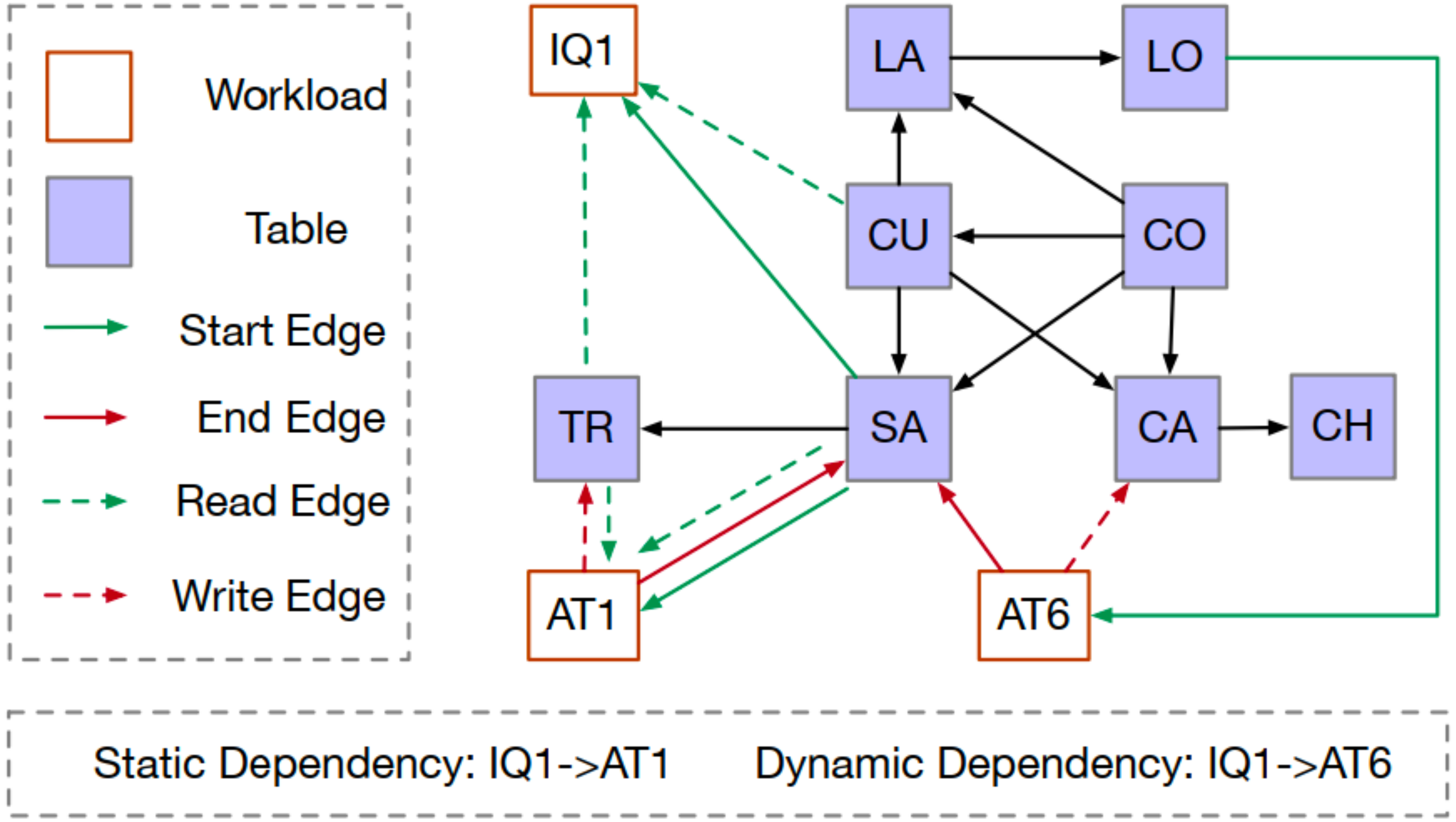
Analytical Transaction (AT)

```
SELECT tr.timestamp, cu.custId
FROM Transfer tr, Customer cu
WHERE tr.sourceId=@blockedId
AND tr.targetId=cu.custId
UNION # Union the transfers
SELECT tr.timestamp, cu.custId
FROM Transfer tr, Customer cu
WHERE tr.targetId=@blockedId
AND tr.sourceId=cu.custId
ORDER BY tr.timestamp LIMIT 10
```

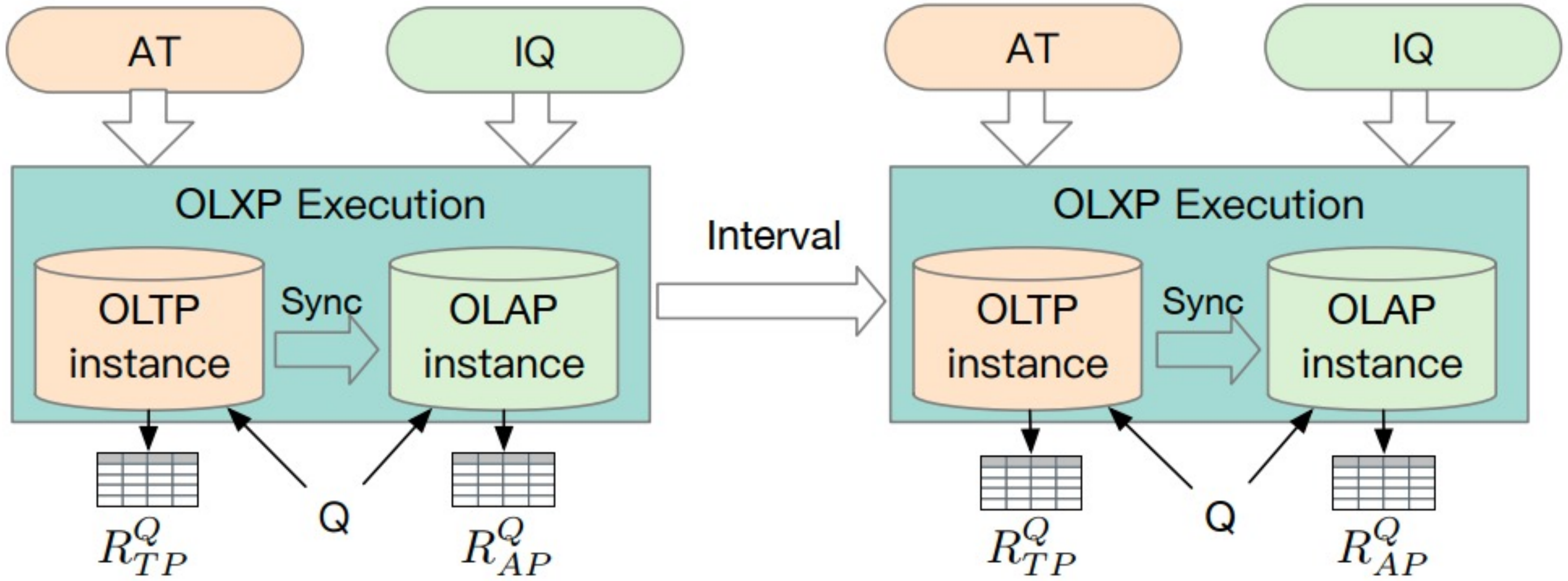
Interactive Query (IQ)

Workload Control

- **Step 1:** for each IQ, build a dependency graph, which depicts their source tables by start edges, read tables by read edges, and write tables by write edges
- **Step 2:** given a risk rate $0 < \alpha < 1$, the IQs and the ATs with static dependency samples the anomalies parameters with the probability α
- **Step 3:** If the target distribution is latest, dependent IQs and ATs operate the same queue. Otherwise, uniform or power-law is used



Freshness Evaluation



□ The Case of Update：

–TP:{（i1,ts1), (i2,ts2)}

–AP:{（i1,ts3), (i2,ts4)}

□ The Case of Insert：

–TP:{（i1,ts1), (i2,ts2), (i3,ts3)}

–AP:{（i1,ts4), (i2,ts5)}

□The Case of Delete：

–TP:{（i1,ts4), (i2,ts5)

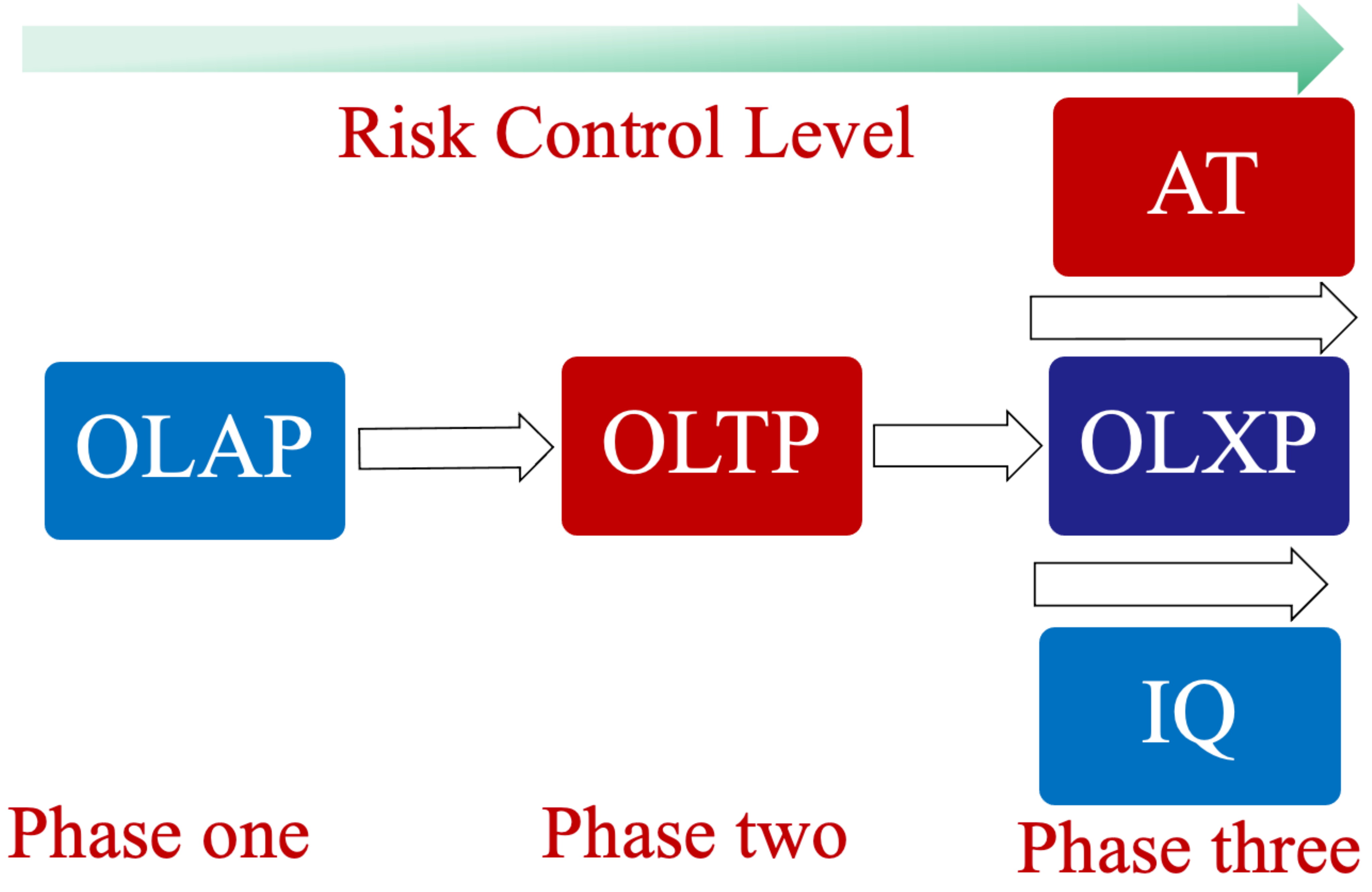
–AP:{（i1,ts1), (i2,ts2), (i3,ts3)}

$$\overline{f_s} = max\begin{cases} ts_{r_a} - ts_{r_b} | r_a \in R^Q_{TP}, r_b \in R^Q_{AP}, i_{r_a} = i_{r_b} & update \\ ts_Q - ts_{r_a} | r_a \in R^Q_{TP}, r_b \notin R^Q_{AP}, i_{r_a} = i_{r_b} & insert \\ ts_Q - ts_{r_a} | r_a \notin R^Q_{TP}, r_b \in R^Q_{AP}, i_{r_a} = i_{r_b} & delete \end{cases}$$

Overall Evaluation

Given the concurrency number (n,m) and the dataset with scale factor SF

- ◻ Phase 1: The first phase executes the OLAP workload with m streams, and each stream contains all the 13 queries with a random order
- ◻ Phase 2: The second phase performs the OLTP workload with n streams
- ◻ Phase 3: The third phase executes the OLXP workload by running the ATs and IQs concurrently, and they are running with n and m workers, respectively.



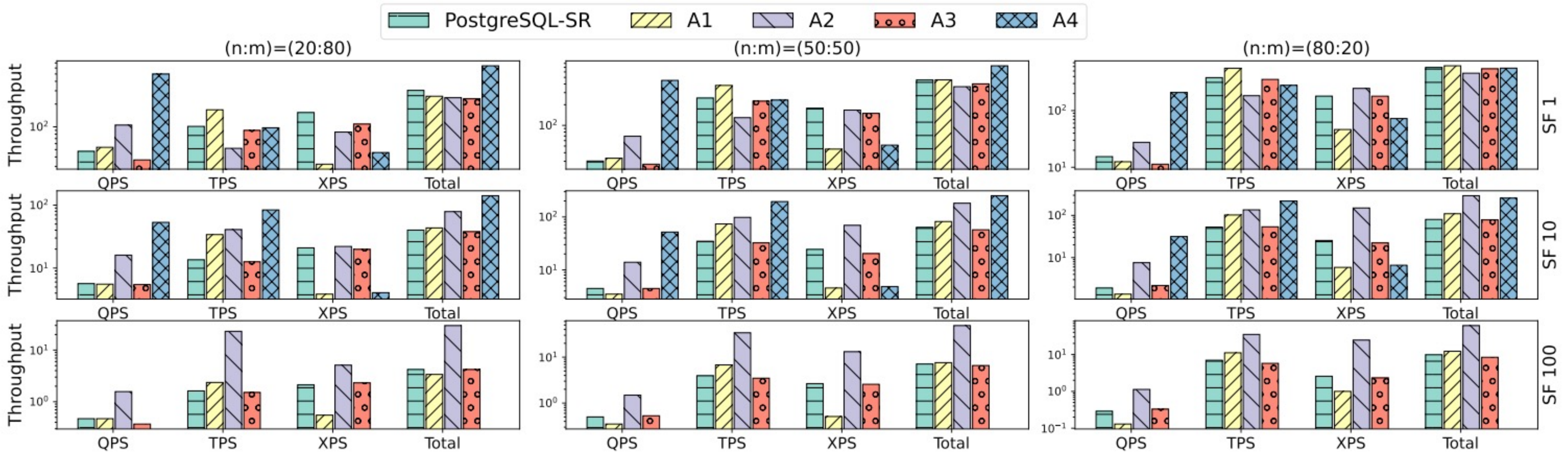
$$\text{H-Score} = SF * \frac{\sqrt[3]{TPS * QPS * XPS}}{\overline{f_s} + 1}$$

Table of Content

Experimental Settings

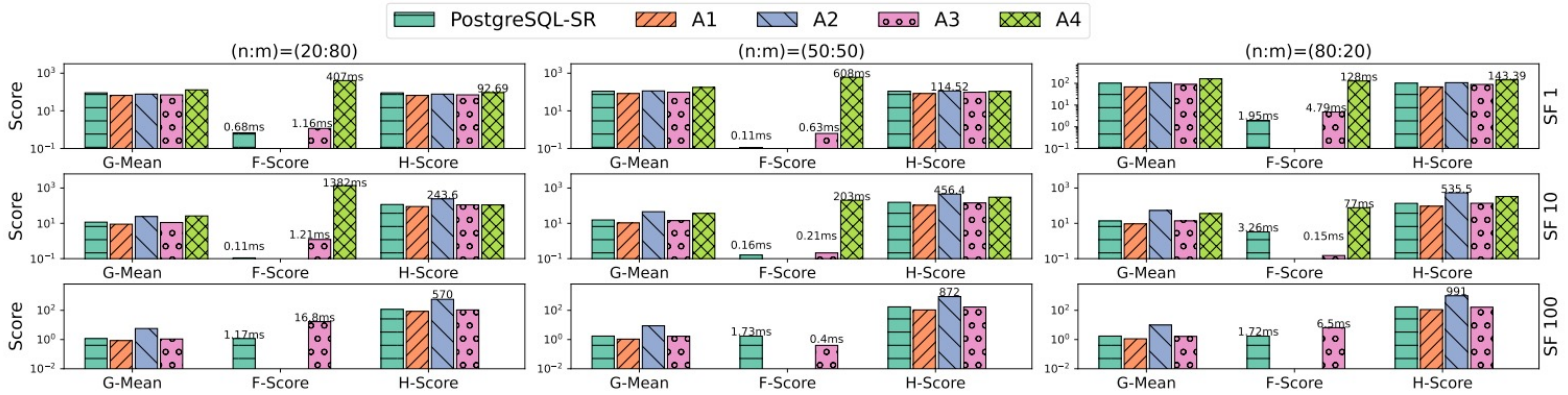
- Environment: four-node cluster, each node has a 2.2 GHz Intel Xeon(R) with 40 physical cores, 128GB RAM
- Dataset: SF1（3+3）, SF10（3+6）, SF100（6+9）
- Concurrency: (n,m) ∈ {(50:50, (20:80), (80:20))}
- SUTs: PostgreSQL 14, A1, A2, A3, A4 (Dewitt Clause)

Overall Throughput



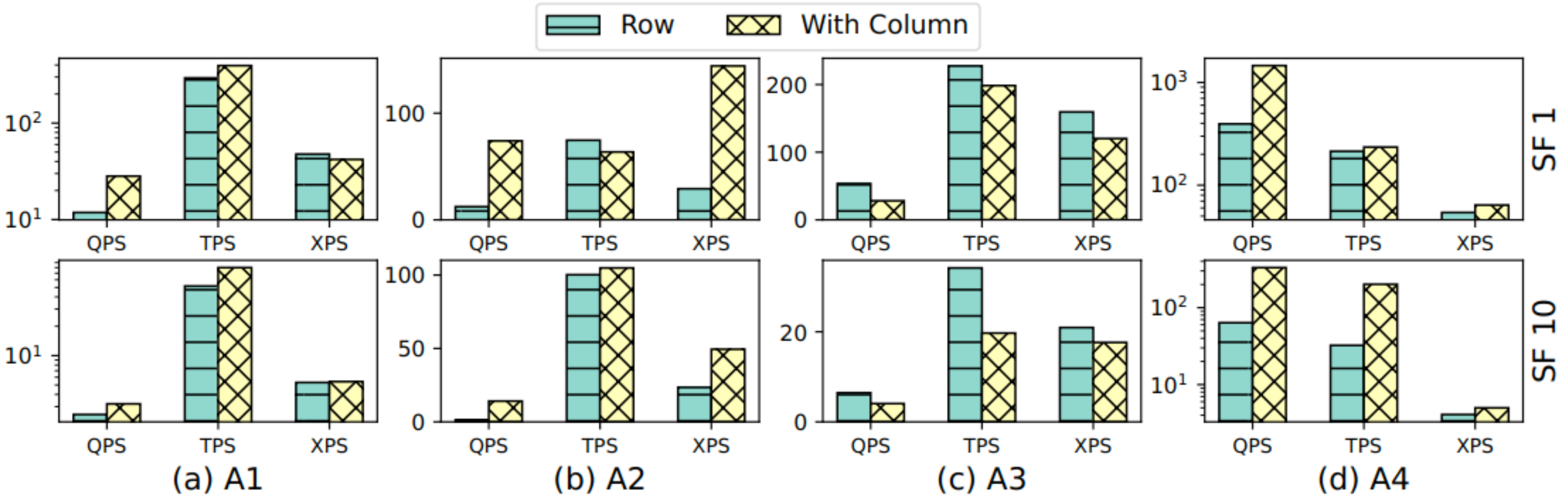
- **Finding 1:** **different SUTs have different pros and cons , e.g., A4 has high QPS, but its XPS is low; A1 has high TPS, but its XPS is low; PG has high XPS, but its QPS is low**
- **Finding 2:** **as the concurrency varies, the total throughput may be different, from (n:m)= 20:80 to (n:m)= 80:20 , A1 and A4 have different total throughput**

Overall Performance



- **Finding 3:** **different SUTs delivers different F-Score, hence the H-Score is different, e.g., A2 has zero freshness and its H-Score is high.**
- **Finding 4:** **Freshness (F-Score) has an impact on the overall performance, e.g., A4 has an F-Score of 608ms, thus its H-Score is lower than A2**

Performance with hybrid row/column



- **Finding 5:** **adding column store generally improves the performance of QPS, TPS, and XPS.**
- **Finding 6:** **A3 has a worse QPS with column store, indicating its hybrid scan may degrade the query performance**

Table of Content

Summary

- Contribution 1: We propose a new benchmark for HTAP databases based on an HTAP-native application.
- Contribution 2: We propose a hybrid workload of OLTP, OLAP, and OLXP, based on the choke-point design.
- Contribution 3: We design a unified metric to make a holistic evaluation of HTAP databases, including freshness and throughput.

Q&A

Thanks！

Question?

HyBench: https://github.com/Rucchao/HyBench-2023/tree/master

Chao Zhang’s Homepage: https://rucchao.github.io/

Email: cyccha6535@gmail.com