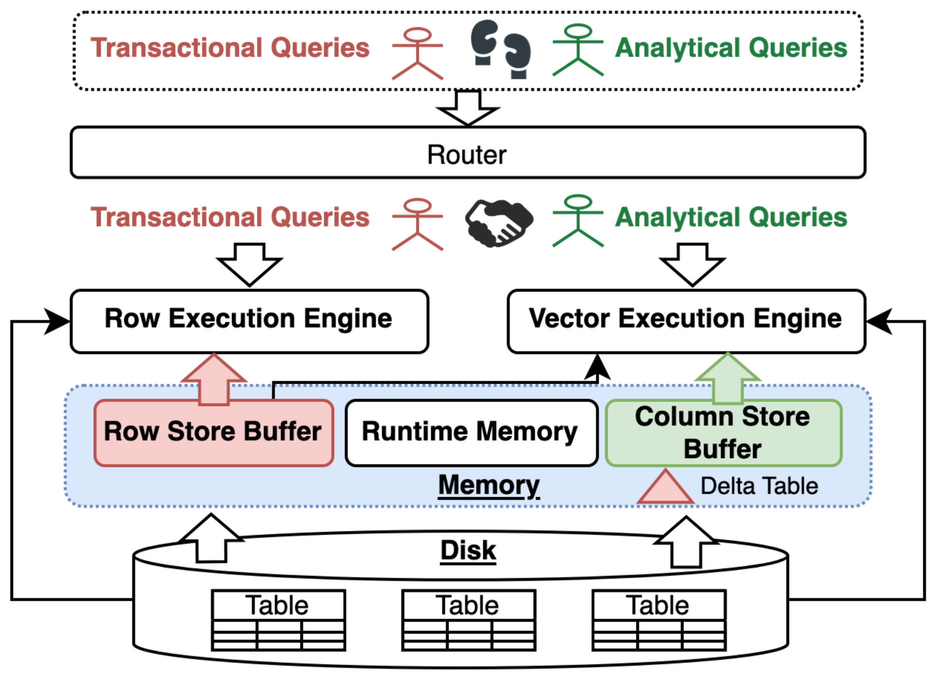
Twisted Twin: A Collaborative and Competitive Memory Management Approach in HTAP Systems

Jiani Yang, Sai Wu*, Yong Wang, Dongxiang Zhang, Yifei Liu, Xiu Tang, Gang Chen

Zhejiang University, Huawei

Introduction

HTAP System v.s. OLTP system



Query Execution

- Add vectorized engine for OLAP
- Support row/column storage reads

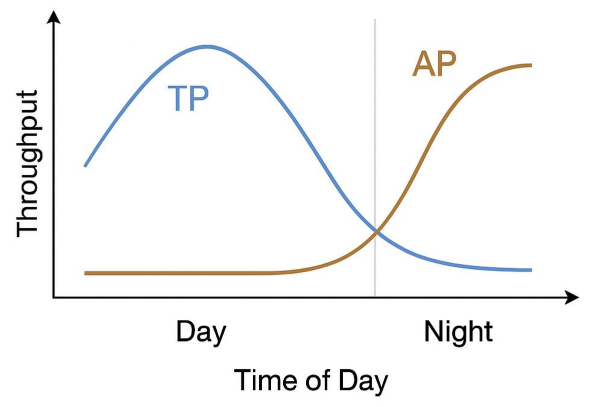
Resource Scheduling

- Shared buffer split into row/column

Data Synchronization

- Use Delta Table to store updates and periodically sync with column store.

Tidal Workload reveals opportunity



Real workloads show a clear **tidal pattern**:

- Daytime → OLTP dominates (transactions, risk checks)
- Nighttime → OLAP dominates (batch jobs, reports)

Problems

Question 1: What data should be loaded into the column store?

Question 2: How to allocate limited buffer pool between row and column store?

Question 3: How to choose an appropriate frequency for data synchronization, and how update costs be considered during column selection?

Question 4: How to smartly rebalance resources to boost overall utilization?

- **Real-time response**
- **Adaptive adjustment**

Large problem search space | Highly coupled Components

Limitations of Current Approaches

Relying on experienced DBAs for manual tuning

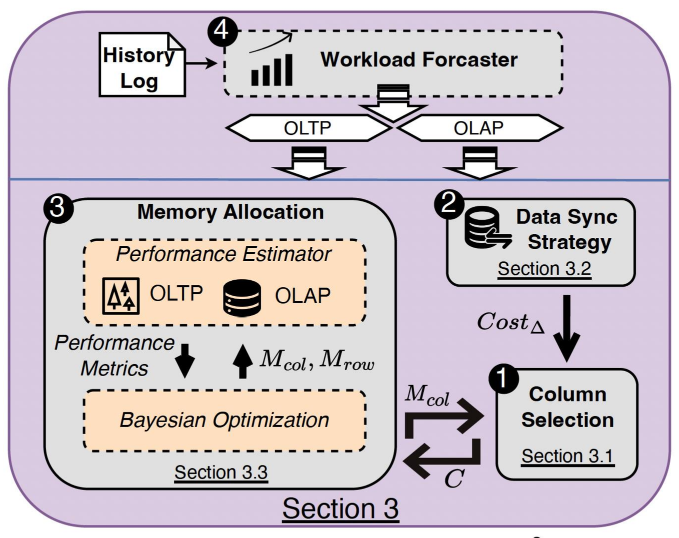
```
ALTER TABLE customer
COLVIEW(companyid, name);
SET ROW_STORE_BUFFER = 80G;
SET COLUMN_STORE_BUFFER = 20G;
```

- **Experience-based, suboptimal**
- **Costly and time-consuming**
- **Poor adaptability to workload shifts**

AUTODRIVEN HTAP SYSTEM

Methodology

Overview



Auto Memory Management T^2

Optimization Goal

- Ensure OLTP throughput as the first priority, and maximize OLAP performance on top of it

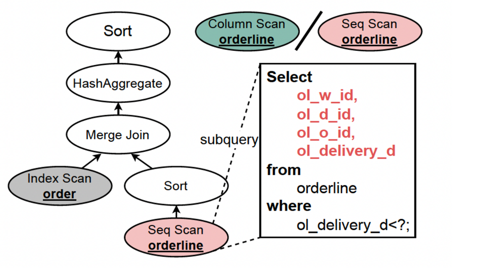
Two-Phase Strategy

- Phase 1: Static configuration
- Phase 2: Dynamic tuning (invokes static configuration when needed)

Column Selection

Column Selection → Knapsack Problem Variant

- Limited memory space → knapsack with limited capacity
- Reading from column store is cheaper than row store → item value
- A query can use column-store scan only if all its required columns are in memory → knapsack with dependency constraints



Problem Formulation:

- Nonlinear Integer Programming Problem

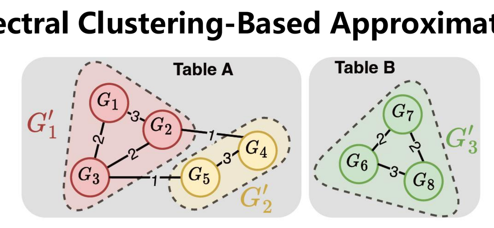
$$\text{Maximize} \quad \sum_{l\in K} \Pi_{i\in G_l} p_l x_i f_l$$
$$\text{Subject to} \quad \sum_{i=1}^{M} w_i x_i \le M_{col}, \quad x_i \in \{0,1\}, \quad i \in M.$$

Nonlinear Problem -> Linear Problem:

- New variable z_l to replace nonlinear terms:

$$\text{Maximize} \quad \sum_{l\in K} p_l z_l f_l$$
$$\text{Subject to} \quad \sum_{i=1}^{M} w_i x_i \le M_{col}, \quad \sum_{i\in G_l} x_i \ge |G_l| \cdot z_l, \quad l \in K, \quad x_i, z_l \in \{0,1\}, \quad i \in M, l \in K.$$

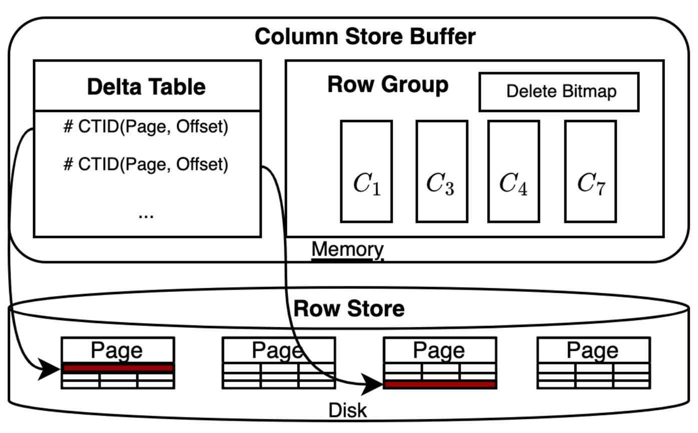
Spectral Clustering-Based Approximation



- Solution: Graph modeling + clustering compression.
- Result: Variables reduced from K to K'.

Data Synchronization

Illustration of Data Synchronization



- **Trade-off:** Frequent synchronization increases overhead, while infrequent synchronization degrades OLAP query performance.
- **Column Selection Impact:** Columns with excessive synchronization costs should be excluded despite potential benefits.

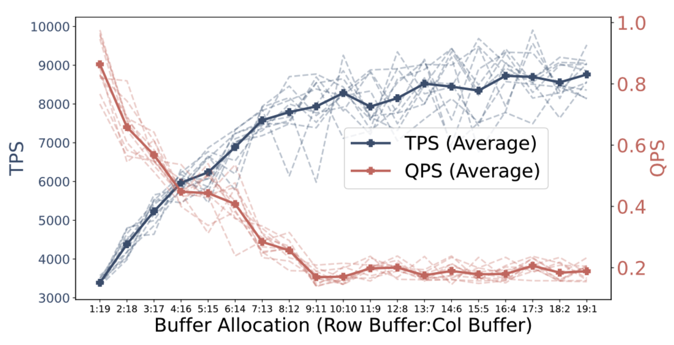
Cost-Model Based Data Synchronization Strategy

- Data updates are uniformly distributed over $[0,T]$, with a total of b updates.
- A synchronization is triggered whenever the number of records reaches the threshold α , leading to a total of $\frac{b}{\alpha}$ synchronizations
- Building a cost model for read/sync: $Cost_{read}(t) = w_0 N(t) + w_1$; $Cost_{sync}(t) = w_2 N(t) + w_3$
- Total cost is a convex function: $Cost\Delta = \frac{b}{\alpha}(w2\alpha+w3)+v(\frac{w0\alpha}{2} + w1)$
- Taking the derivative of the total cost function and setting it to zero: $\frac{dCost\Delta}{d\alpha} = -\frac{bw_3}{\alpha^2} + \frac{vw_0}{2} = 0$
- Solving for the optimal threshold: $\alpha* = \sqrt{\frac{2bw_3}{vw_0}}$

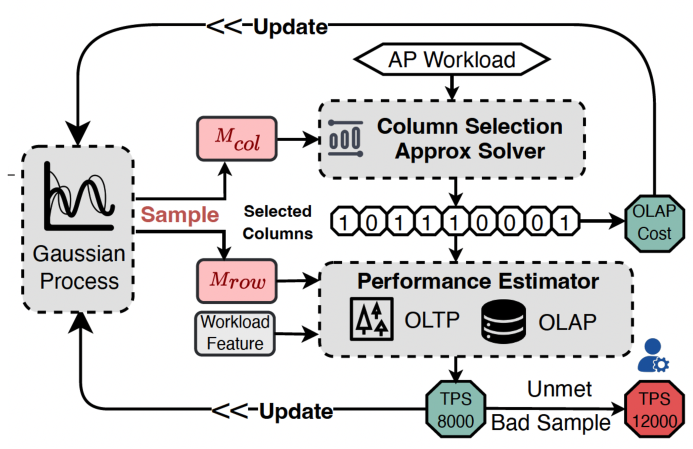
Memory Allocation

The Impact of Buffer Allocation Between Row and Column

- OLTP: More memory helps cache hot data, reducing I/O and improving TPS.
- OLAP: More memory keeps more columns in memory, minimizing I/O from row store access.



Bayesian-based Memory Allocation



Bayesian Optimization Workflow

1. Collect samples & costs
2. Build surrogate model
3. Select candidate (e.g., EI)
4. Evaluate & update

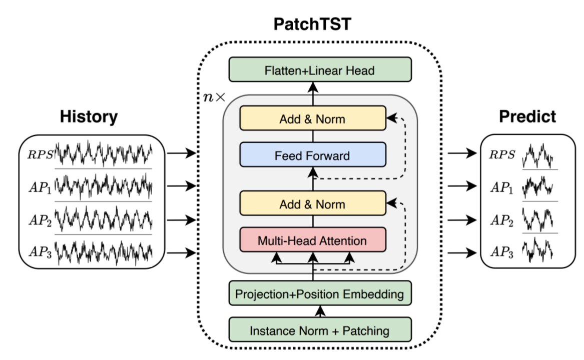
$$cost = \begin{cases} Cost_{OLAP} & ,when\ P_{OLTP} > \theta \\ +\infty & ,when\ P_{OLTP} < \theta \end{cases}$$

5. Repeat the iteration until convergence, and finally obtain the optimal allocation

Dynamic Tuning

Predict Future Workloads, Plan Ahead

- Use historical load patterns to forecast future trends



Trigger Timing Selection

- Taking the cost of column loading into account, how to determine the appropriate timing to trigger a static algorithm for memory reallocation?
- We propose a greedy algorithm: reconfigure when benefits outweigh costs.

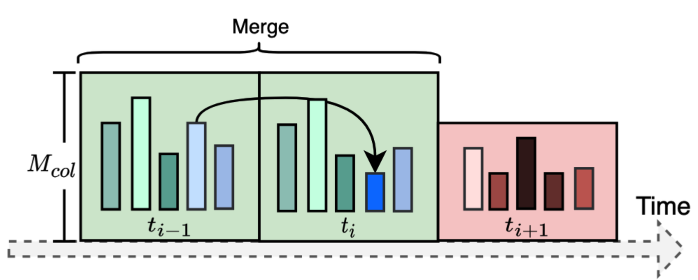


Figure 7: Illustration of Time Interval Merging

Experiments

Metrics

- Metric for measuring OLTP performance

$$FR = \frac{TPS}{RPS} \quad \begin{array}{l}\text{(System Throughput)} \\ \text{(User Demand)}\end{array}$$

- Metric for measuring OLAP performance

$$Impr = (T_{GaussDB} - T_{GaussDB-HTAP})/T_{GaussDB} \times 100\%$$

Performance improvement compared to the original GaussDB (row-store only)

Benchmark Construction

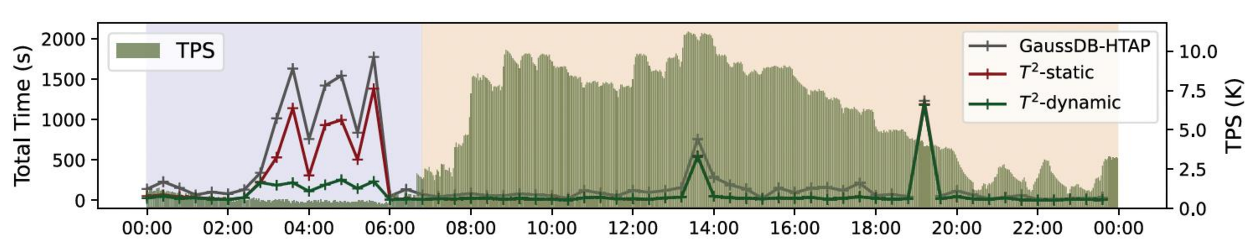
- Query arrival patterns + synthetic benchmarks (ChBenchmark, HyBench)
- OLTP/OLAP query rates extracted from anonymized logs

Overall Performance

HyBench(M_{total} = 80G)

Method	Static-7:1		Static-1:1		Static-1:7		STMM		T^2-static		T^2-dynamic	
	FR	Impr	FR	Impr	FR	Impr	FR	Impr	FR	Impr	FR	Impr
HAMCS	1.00	13.56%	1.00	22.20%	0.912	23.11%	1.00	27.95%	1.00	28.06%	1.00	35.88%
IPNC	1.00	15.74%	1.00	32.12%	0.936	39.41%	1.00	34.98%	1.00	36.45%	1.00	48.70%
GACC	1.00	12.03%	1.00	40.62%	0.940	44.29%	1.00	39.38%	1.00	42.49%	1.00	50.28%
T^2-CS-Approx	1.00	16.39%	1.00	49.33%	0.953	60.03%	1.00	46.97%	1.00	49.92%	1.00	65.90%
T^2-CS	1.00	18.13%	1.00	51.36%	0.956	60.69%	1.00	51.17%	1.00	54.44%	1.00	**67.07%**

During a One-Day Simulation



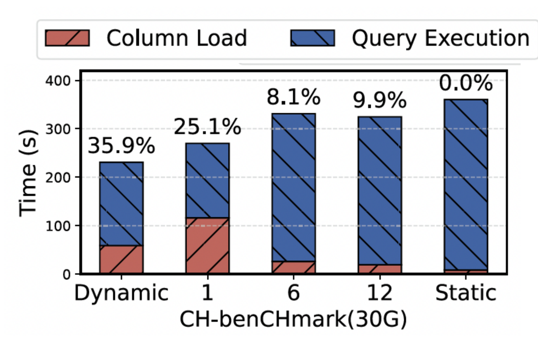
Evaluation with varying memory limitation

- When memory is tight, the model prioritizes OLTP throughput, allocating most memory to the row store and maintaining high FR.

M_{total}		5G	15G	30G	60G
STMM	M_{col}	2.55	7.53	15.01	29.95
	FR	0.59	0.70	0.98	1.00
	Impr	15.98%	20.08%	18.18%	36.78%
T^2-static	M_{col}	0.02	0.11	9.12	32.06
	FR	0.85	0.93	1.00	1.00
	Impr	1.13%	1.64%	16.49%	39.33%
T^2-dynamic	FR	0.86	0.95	1.00	1.00
	Impr	7.78%	11.89%	22.54%	53.69%

Effect of Reallocation Trigger Algorithm

- The percentages in the figure represent the percentage reduction in total time relative to the static algorithm.

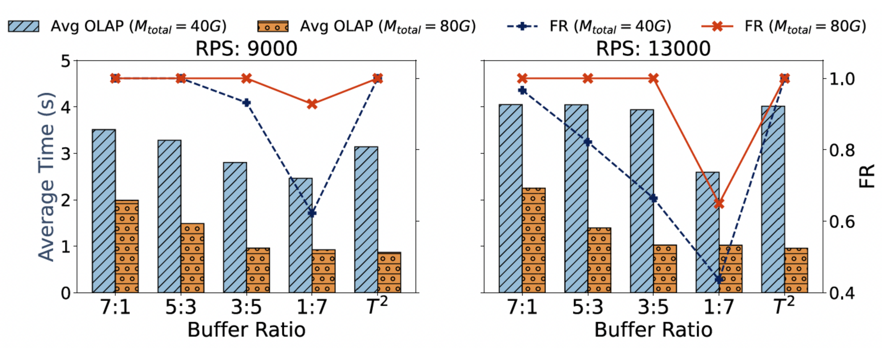


Analysis of Different Column Selection Algorithms

- Although some competing methods occasionally achieve performance similar to that of T^2, they are prone to getting trapped in local optima, leading to a higher max loss.

Method	Hybench		CH-benCHmark	
	Max loss	Max Time	Max loss	Max Time
HAMCS	-41.29%	0.003s	-25.99%	0.00s
IPNC	-8.42%	0.08s	-19.49%	0.17s
GACC	-7.62%	18.17s	-7.17%	8.69s
T^2-CS-Approx	**-2.55%**	10.10s	**-3.49%**	9.06s
T^2-CS	-	45.58s	-	65.93s

Comparative Analysis of T^2 Versus Fixed Ratio Memory Allocation



Contact: jianiyang_cs@zju.edu.cn