FAST '25

HaSiS: A Hardware-assisted Single-index Store for Hybrid Transactional and Analytical Processing

Kecheng Huang[1], Zhaoyan Shen[2], Zili Shao[1], Feng Chen[3] and Tong Zhang[4,5]

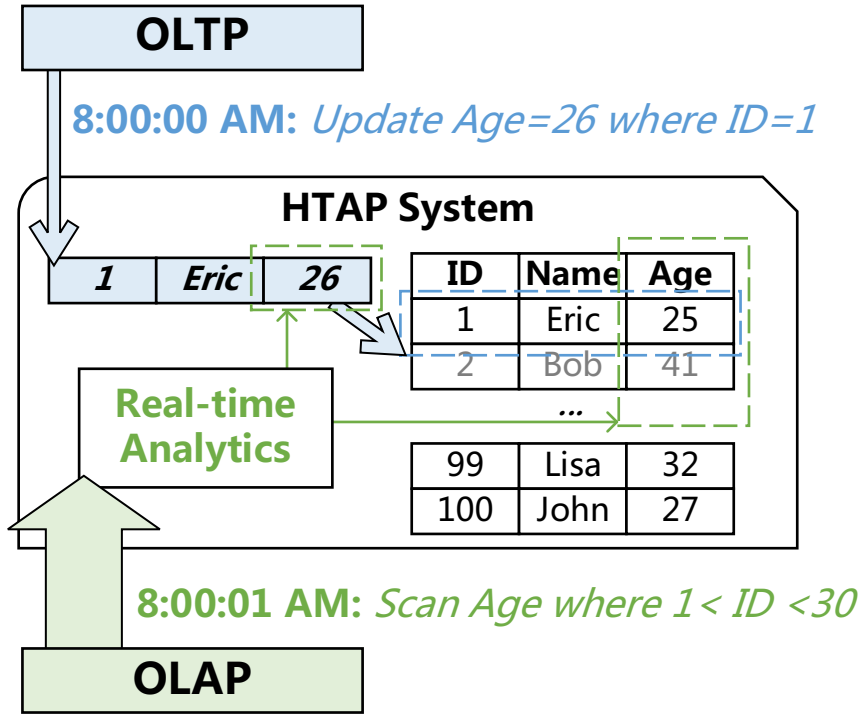
[1]The Chinese University of Hong Kong [2]Shandong University [3]Indiana University Bloomington

[4]Rensselaer Polytechnic Institute [5]ScaleFlux Inc.

Hybrid Transactional and Analytical Processing (HTAP)

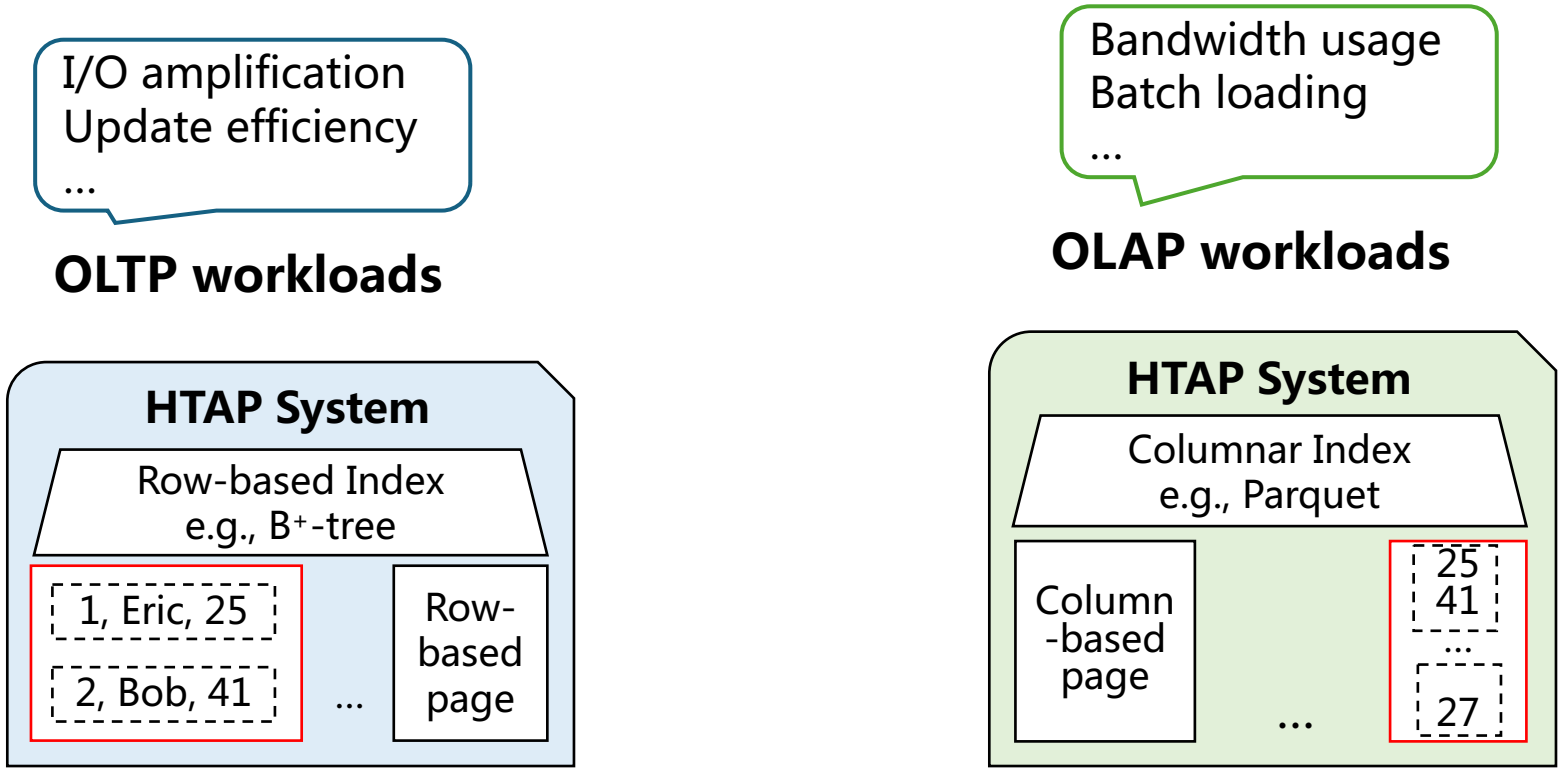
A data processing architecture for simultaneous execution of OLTP and OLAP

- Critical in finance, advertisement, e-commerce, …
- **Key criteria**
  - **Optimal performance** for both
    - **OLTP** (Online Transactional Processing)
      - Small-random record updates/searches
    - **OLAP** (Online Analytical Processing)
      - Large-scale attribute (column) scans
  - **Real-time data analytics**



Well-known Conflict of Optimal Storage Formats

- **OLTP prioritizes row-based storage format**
- **OLAP favors column-based storage format**



Multi-Index HTAP Design

Existing HTAP systems primarily follow a multi-index design strategy

- Mixing row-based index and columnar index
- Realizing data synchronization via runtime **cross-index-domain data migration**
- **Design Approaches**
  - **Multi-index, multi-store:** TiDB[1], F1 Lighting[2], PolarDB-IMCI[3], …
  - **Multi-index, single-store**: SAP HANA[4], Hyper[5], MemSQL[6], …

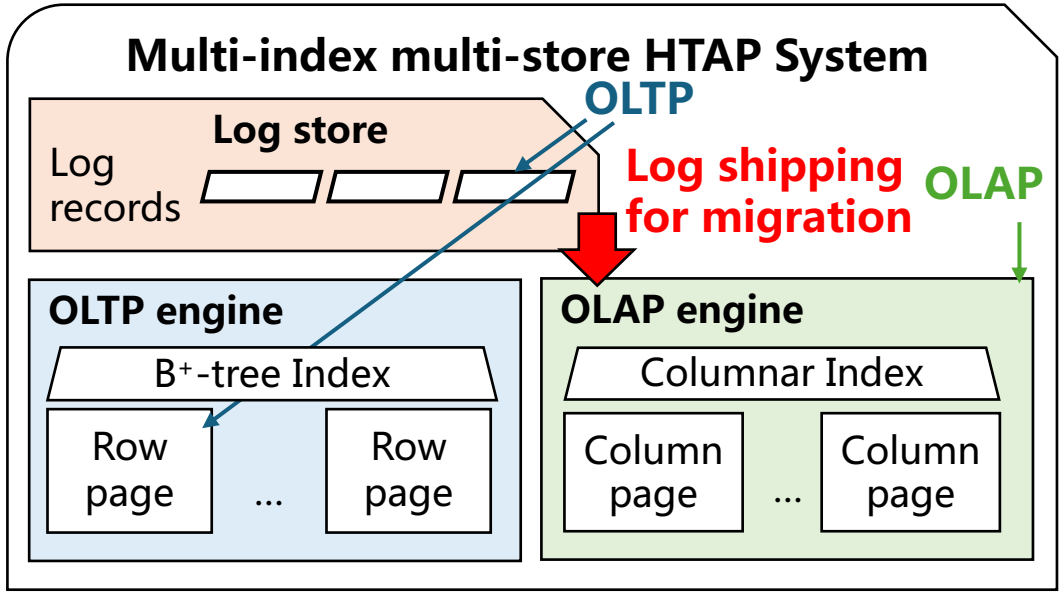
1. https://docs.pingcap.com/tidb/stable
2. https://dl.acm.org/doi/10.14778/3415478.3415553
3. https://help.aliyun.com/zh/polardb/polardb-for-mysql/release-notes-for-imcis
4. https://www.sap.com/index.html
5. https://hyper-db.de/
6. https://www.singlestore.com/

Multi-Index, Multi-Store Design

Existing multi-index, multi-store design

- An OLTP engine (row-based index) for transactional updates
- A OLAP engine (column-based index) for analytical queries
- Cross-domain data migration: converting row-based data into a column-based format
- **Limitations**
  - **Resource-consuming data movement**
  - **Data freshness problem**

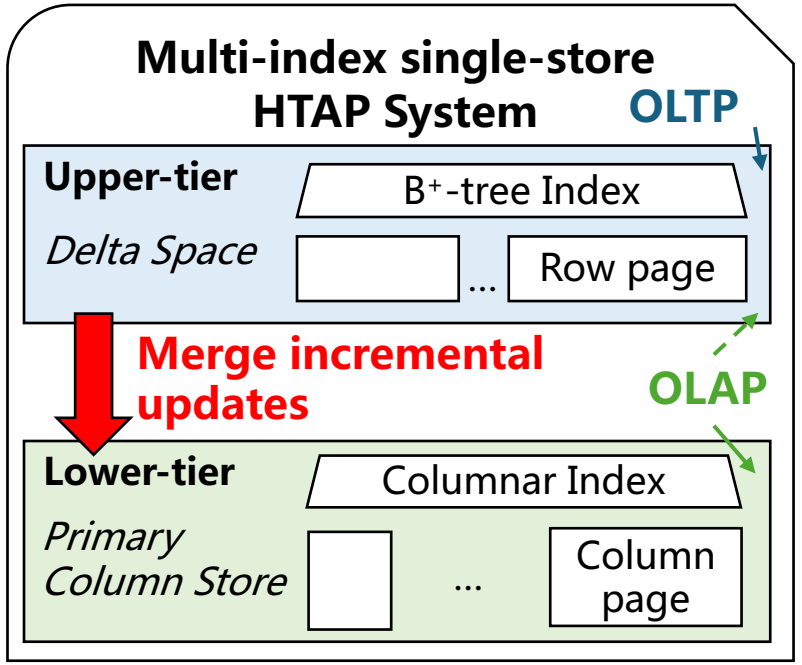
Column store may not be updated in time as transactional updates are temporarily held in the log!
➢ **20ms~8min wait for OLAP scan**



Multi-Index, Single-Store Design

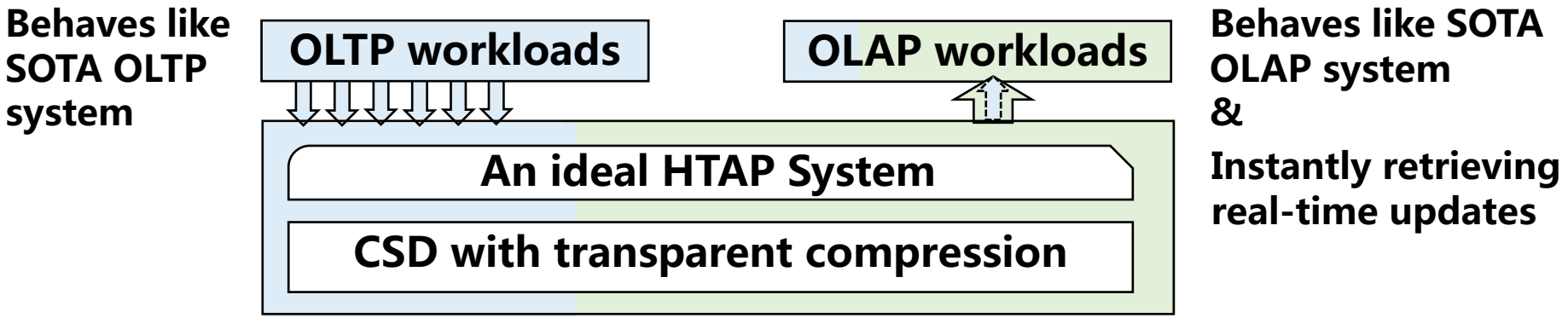
Existing multi-index, single-store design

- Consolidating row-/column-based data within a single system
  - A row-based delta tier atop the main column-based store to enhance an OLAP systems for supporting OLTP workloads
  - Row deltas are periodically merged into the columnar data in the background
- **Limitations**
  - **Index and read amplification of OLAP query**
    - Both data sources should be accessed
  - **Severe performance degradation**
    - When facing larger-than-memory datasets



Design Goal

- **Problem**: Data in existing designs are separately managed in distinct index domains
  - The **cross-index data migration** introduces significant overheads
- **Challenge:** How to simultaneously achieve:
  - ➢ **Optimal OLTP performance**
  - ➢ **Optimal OLAP performance**
  - ➢ **Instant OLAP data freshness**
- **A key observation:** Leveraging **new storage hardware**, other than relying on multiple distinct indexes to reconcile the OLTP vs. OLAP storage format conflict



CSD with Built-in Transparent Compression

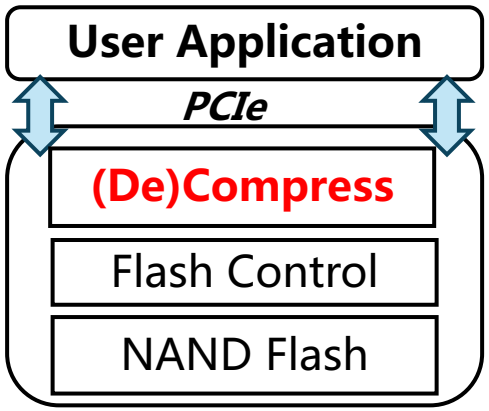
A special CSD (Computational Storage Drives) with data compression functionality

- **Low-cost compression.** It is seamlessly transparent to the host system. The compression engine operates with low latencies **(~5µs)**

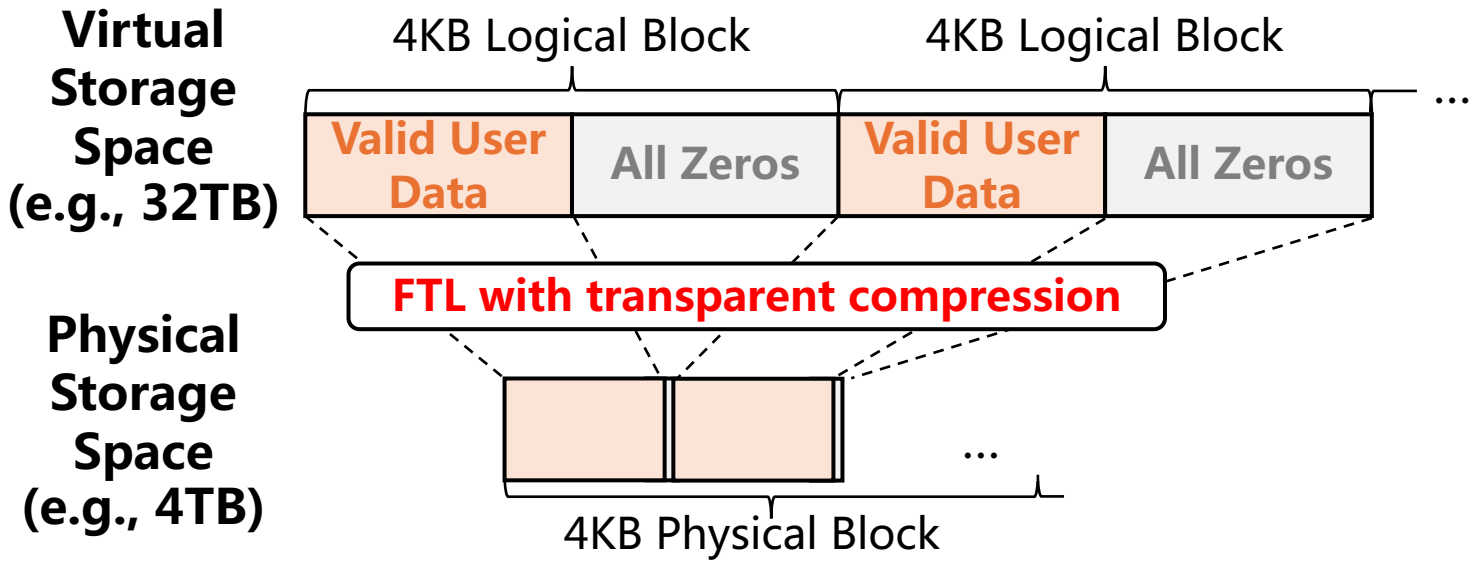
CSD with Built-in Transparent Compression

A special CSD (Computational Storage Drives) with data compression functionality

- **Low-cost compression.** It is seamlessly transparent to the host system. The compression engine operates with low latencies **(~5µs)**
- **Virtualized logical storage space enables sparse data management.** 4KB LBA blocks (e.g., 1KB real data and 3KB zeros) can be partially filled without sacrificing physical space

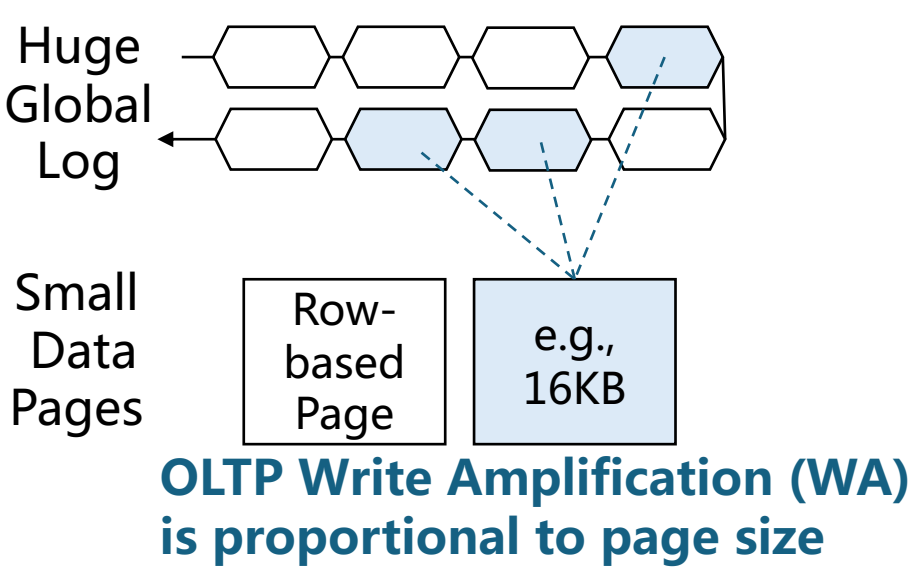


CSD with Transparent Compression

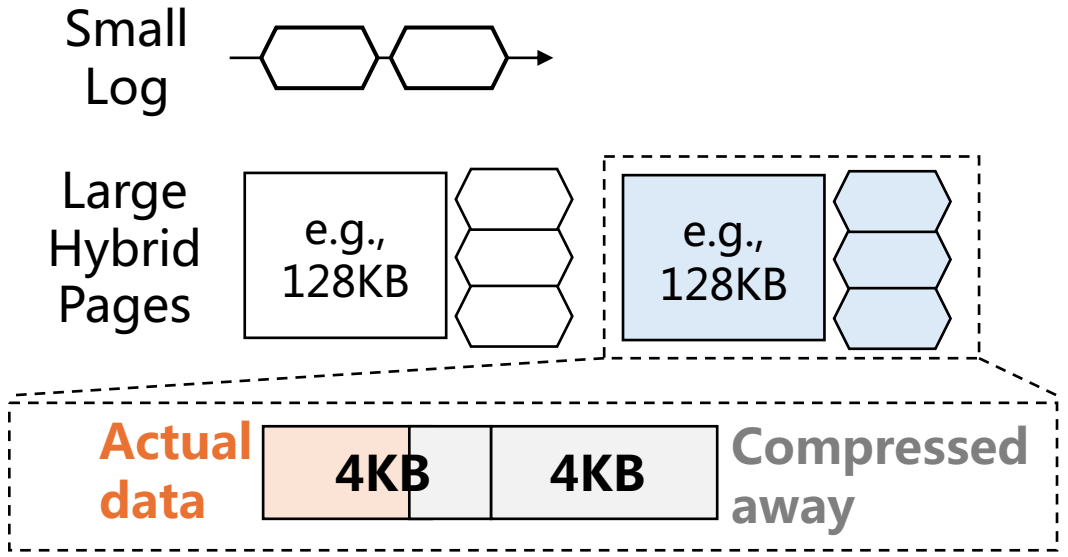


Insight#1: CSD decouples write amplification from page size

- OLTP favors small page for less write amplification (WA) during **in-place update**
- Large log mitigates WA through accumulation but results in **sub-optimal crash recovery**
- CSD enables distributing the huge log as **per-page logs** without storage penalty
  - Simultaneously achieving mitigated WA and better recovery



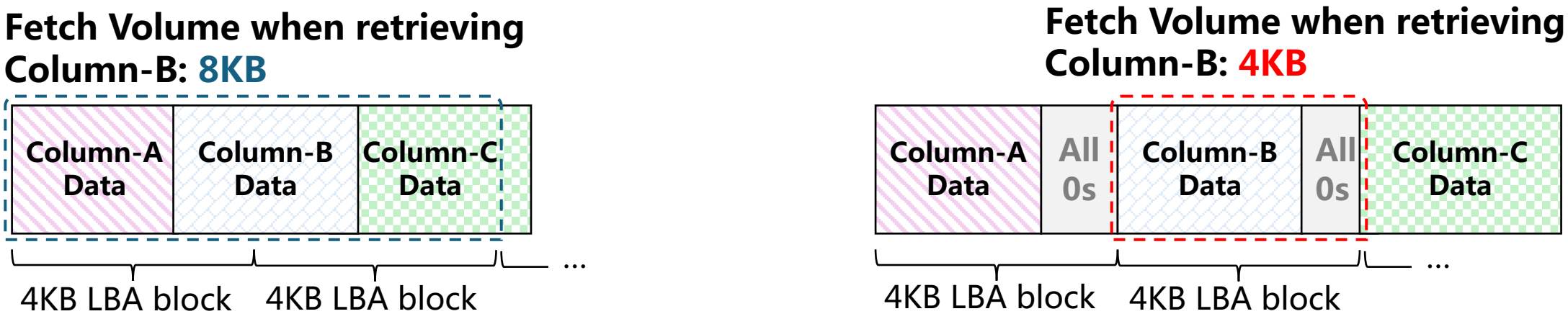
Conventional SSD



CSD with compression

Insight#2: CSD reduces intra-page column read amplification

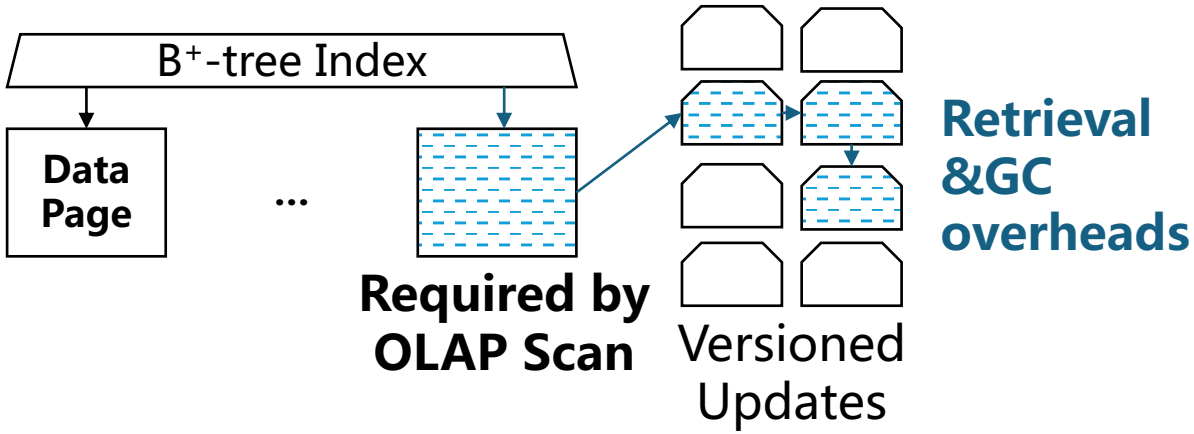
- Columnar page layout is normally condensed for space saving, but results in **read amplification (RA)**
- CSD enables **column-aligned sparse layout** for less RA without extra physical space usage



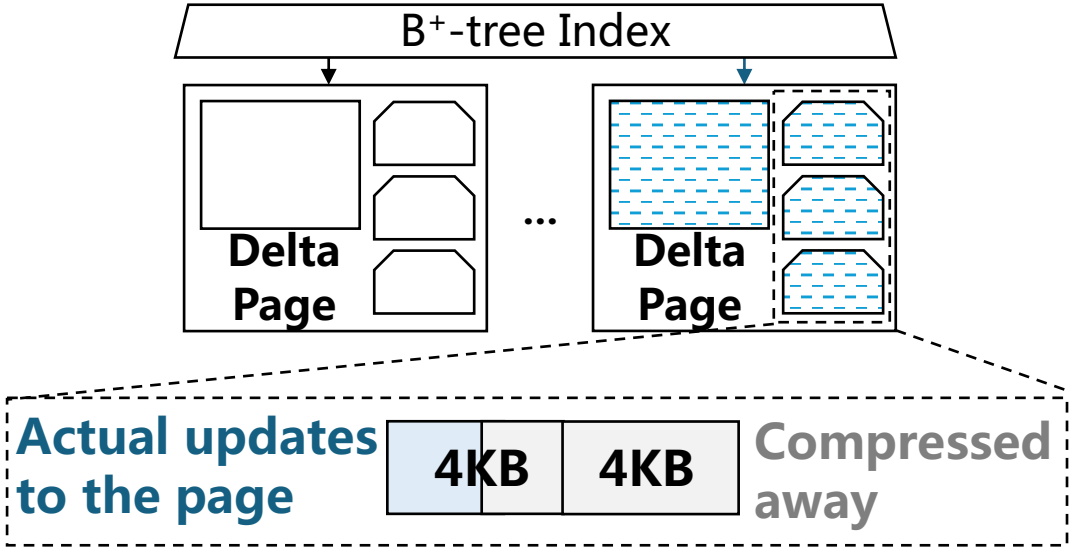
Insight#3: CSD enables per-page storage of multi-version records

- The separate version storage necessitates

  (1) Extra retrieval for OLAP scans (2) Garbage Collection (GC) overheads
- CSD enables **per-page version storage** for efficient OLAP scans and I/O efficient background GC



Conventional SSD

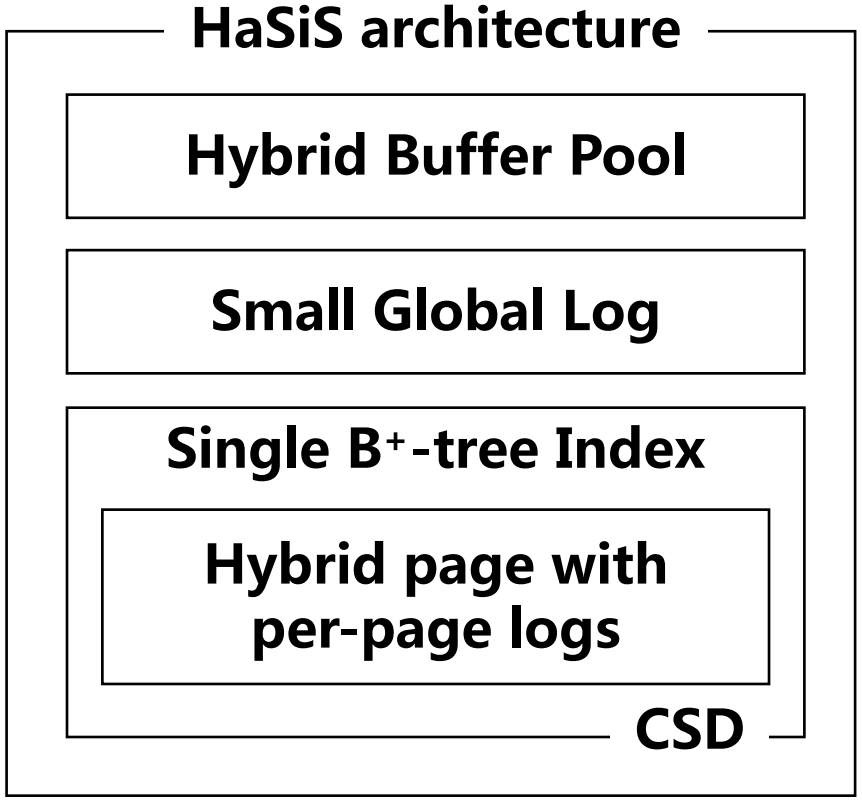


CSD with compression

Our Design: HaSiS (Hardware-assisted Single-index Store)

Design components

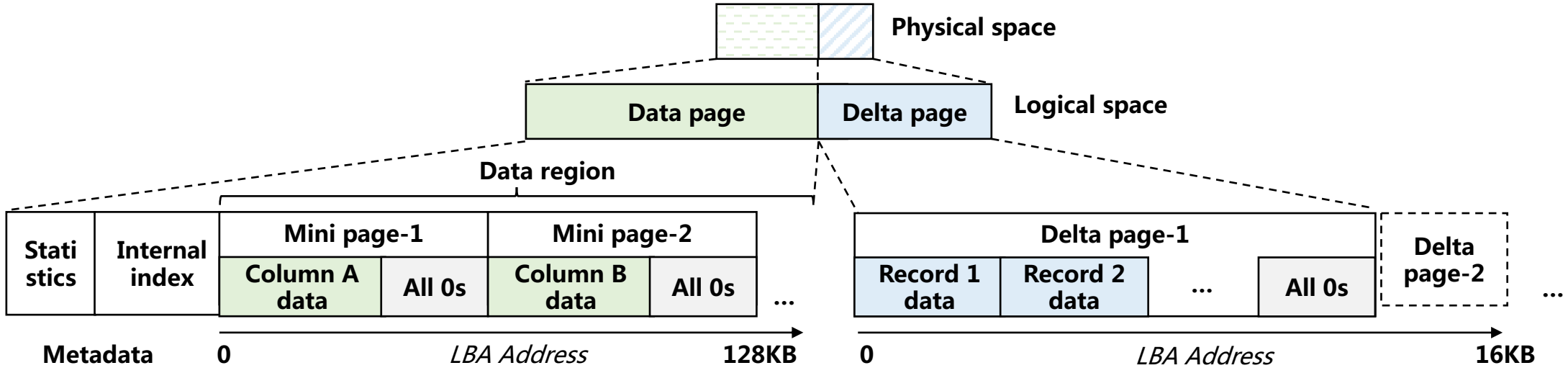
- Hybrid buffer pool
- Small global log
- Hybrid page format with per-page logs



Our Design: HaSiS (Hardware-assisted Single-index Store)

Hybrid page format

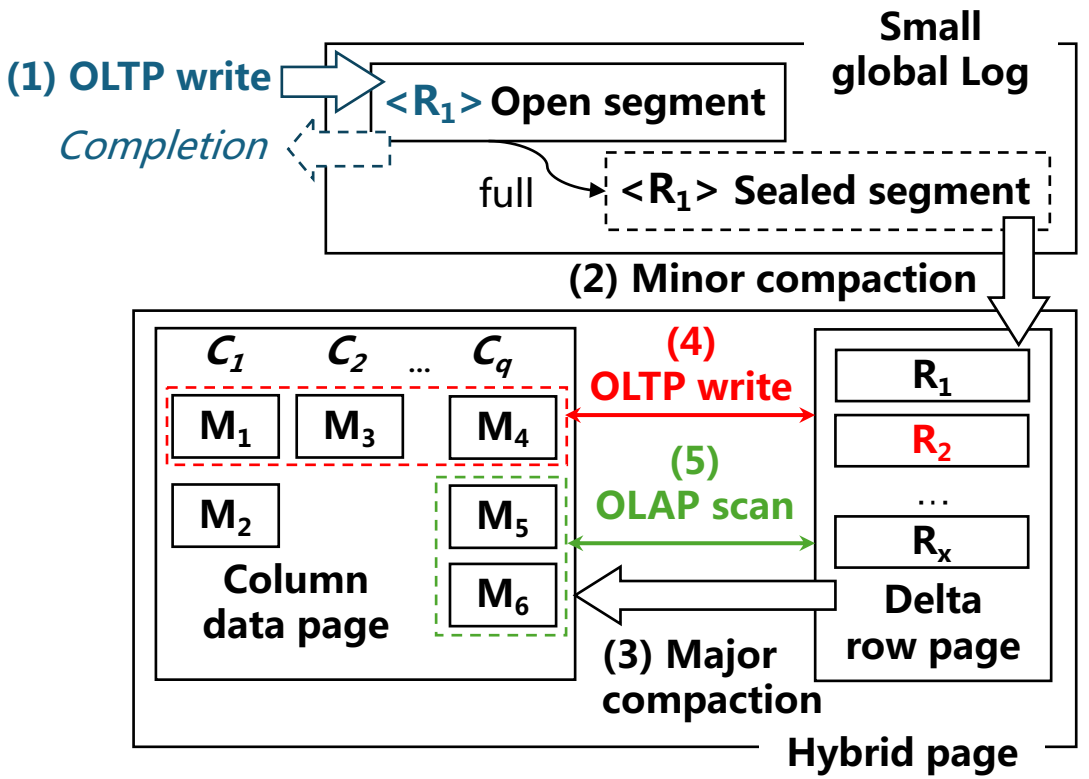
- Largely relaxed B⁺-tree page size (e.g., 128KB +16KB) for both OLTP and OLAP
  - **Column-based data page** for OLAP batch scans
    - Sparsely aligned mini-pages for enhanced I/O efficiency
  - **Row-based delta page** for OLTP updates
    - Efficient OLTP updates through CSD-enabled per-page logging



Our Design: HaSiS (Hardware-assisted Single-index Store)

Log and compaction

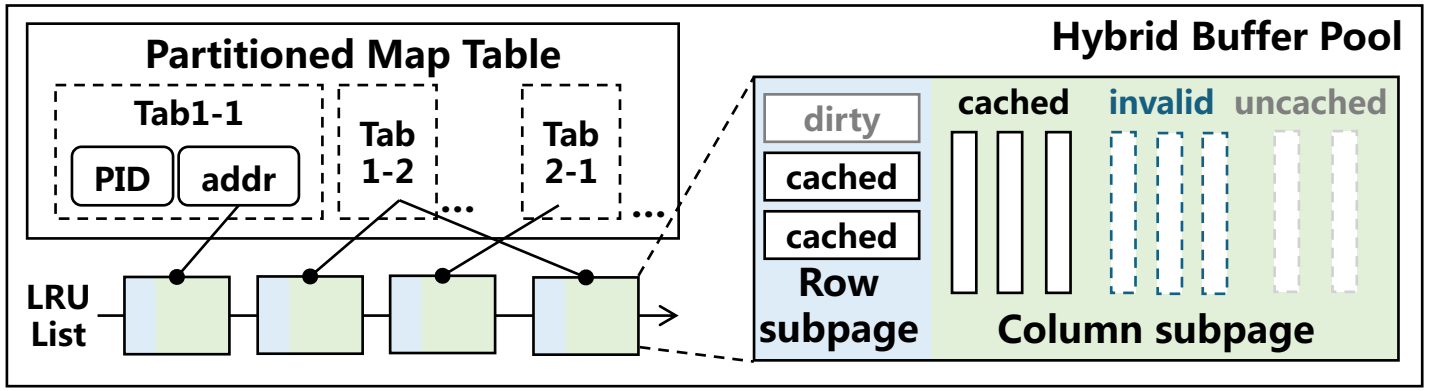
- A small global log for update consolidation
- **Two-phase compaction**
  - Highly parallel batched I/Os
    - Fully utilize CSD high bandwidth
  - **Minor compaction**
    - Batched log-to-row-page merge
  - **Major compaction**
    - Parallelized row-to-column merge



Our Design: HaSiS (Hardware-assisted Single-index Store)

Hybrid buffer pool

- An invalidation-based approach for OLTP/OLAP performance isolation
  - Column subpage is directly invalidated during updates for mitigated rewriting



More details can be found in paper

- Multi-version storage for temporary updates

Evaluation

Implementation

- A **fully functional prototype**[*] of HaSiS
- A **benchmark tool**[*] converts complex SQL queries to simple and table-specific SQL requests
- Comparisons with HTAP (**TiDB**), OLTP (**MySQL** and **PostgreSQL**), OLAP (**Parquet**) and **baseline** Row/Column store designs

Setup

- A server equipped with a 22-core 2.2GHz Intel Xeon E5-2696 v4 CPU, 64GB DDR4 DRAM, and a **7.68TB ScaleFlux CSD-3310 SSD** [@]
  - Already in volume production and widely deployed in data centers globally
- Ubuntu 20.04 LTS with Linux Kernel 5.15 and Ext4 file system

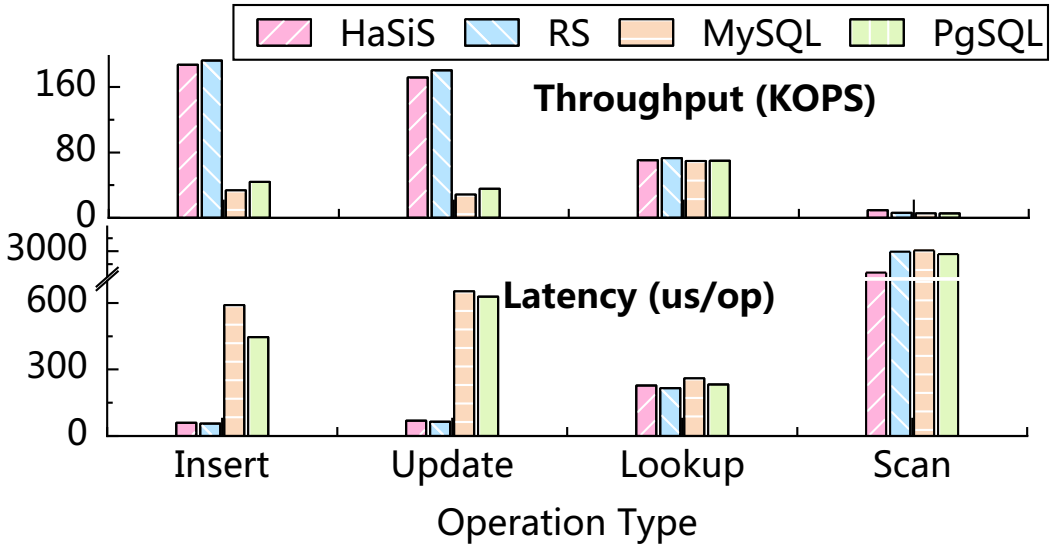
[*] https://github.com/ericaloha/Hasis

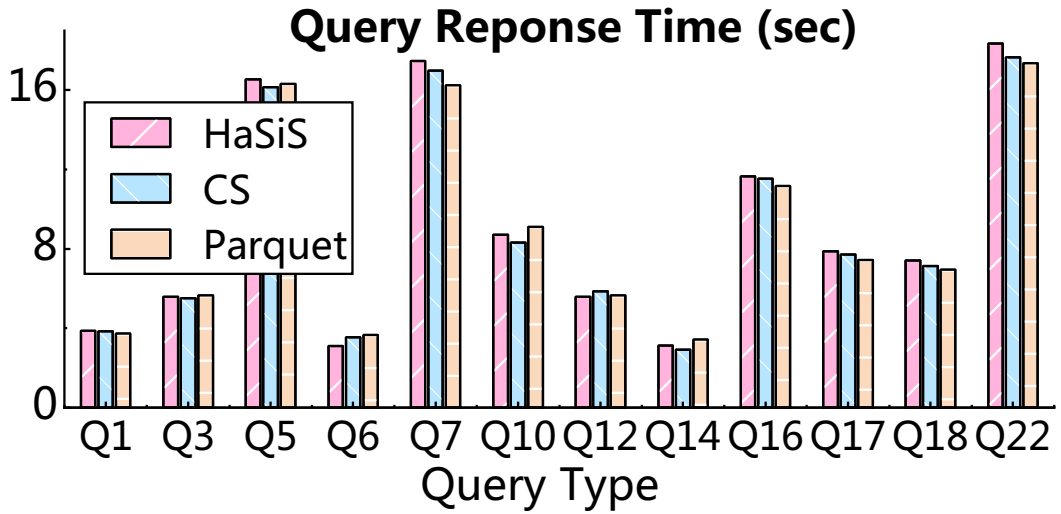
[@] https://scaleflux.com/products/csd-3000/

Evaluation

Comparison with OLTP/OLAP system

- HaSiS shows comparable OLTP throughput and latency comparing with RowStore baseline, MySQL and PostgreSQL
- HaSiS shows competitive OLAP query performance comparing with ColumnStore baseline and Parquet

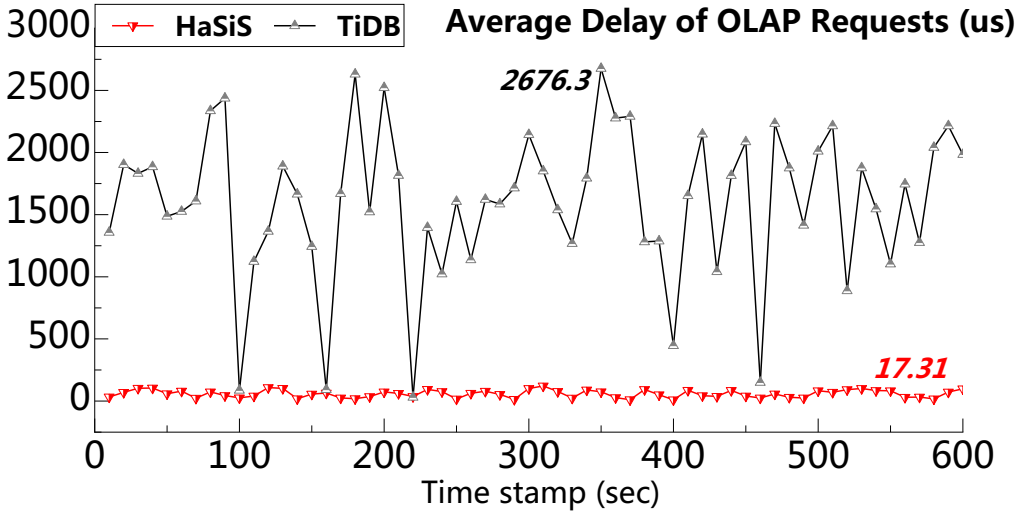


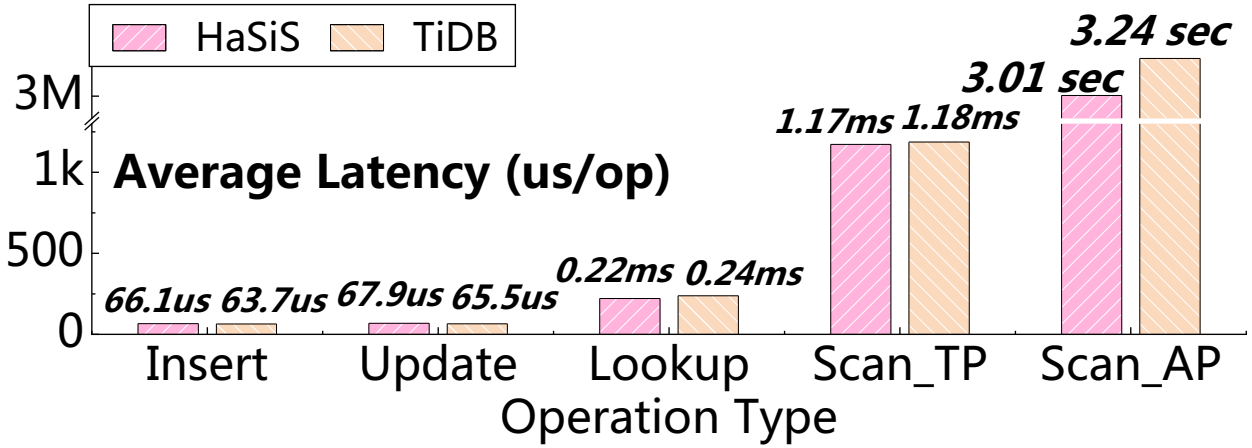


Evaluation

Comparison with HTAP system

- HaSiS exhibits instant and stable OLAP data freshness compared to TiDB
- HaSiS exhibits competitive HTAP performance compared to TiDB

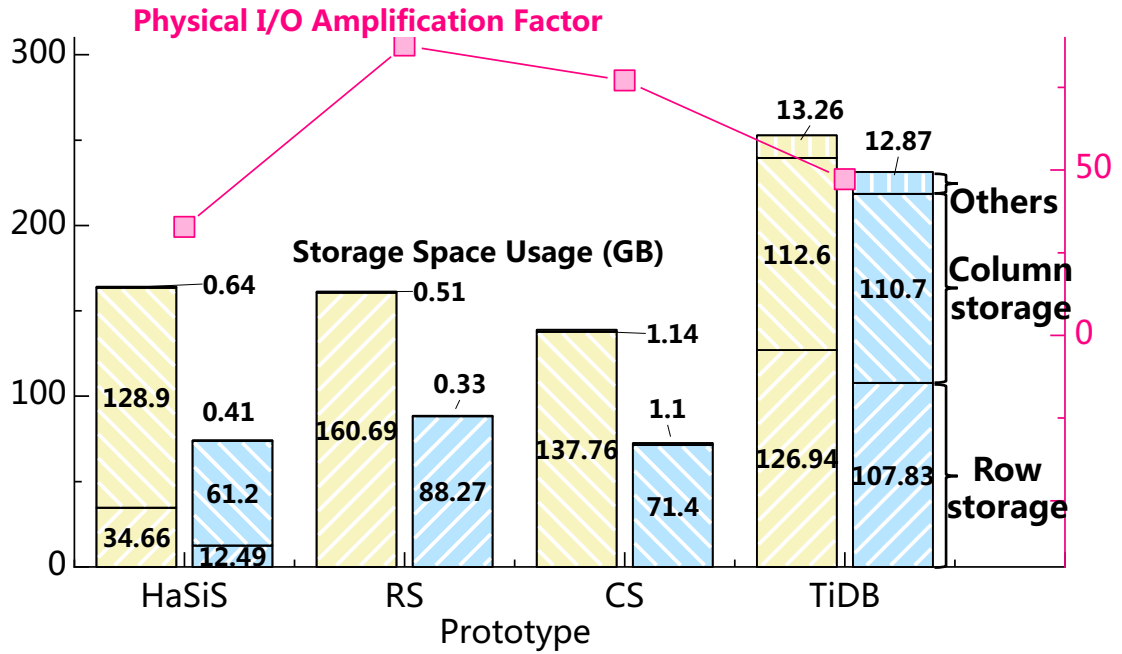




Evaluation

Effects of optimizations

- HaSiS is space-efficient compared to contemporary HTAP system
- HaSiS requires marginal extra physical space usage



Conclusion

- We propose **HaSiS**, a CSD-enabled Single Index HTAP design
- HaSiS addresses the longstanding trade-off of **Data freshness vs. HTAP performance**
- HaSiS leverages CSD to create **sparse data structures** without impacting physical capacity
- We thoroughly evaluate HaSiS on a **commercially available CSD platform**

Thanks!

Q&A

Kecheng HUANG
The Chinese University of Hong Kong
kchuang21@cse.cuhk.edu.hk