Retrofitting High Availability Mechanism to Tame Hybrid Transaction/Analytical Processing

Sijie Shen, Rong Chen, Haibo Chen, Binyu Zang

{ds_ssj, rongchen} @ sjtu.edu.cn

Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University

Shanghai Artificial Intelligence Laboratory

Engineering Research Center for Domain-specific Operating Systems, Ministry of Education, China

Typical workloads of databases

OLTP
(OnLine Transaction Processing)

Short-term read-write transactions

- Online order processing
- Stock exchange
- E-commerce

OLAP
(OnLine Analytical Processing)

Long-term read-only queries

- Business intelligence
- Financial reporting
- Data mining

HTAP: new type of workloads

OLTP **HTAP** **OLAP**

(**H**ybrid **T**ransaction/**A**nalytical **P**rocessing)

Real-time queries on data generated by transactions

- IoT
- Healthcare
- Fraud detection
- Personalized recommendation

Requirements

Performance

- Minimizing performance degradation (e.g., < 10%)
- Millions of transactions per second[1]

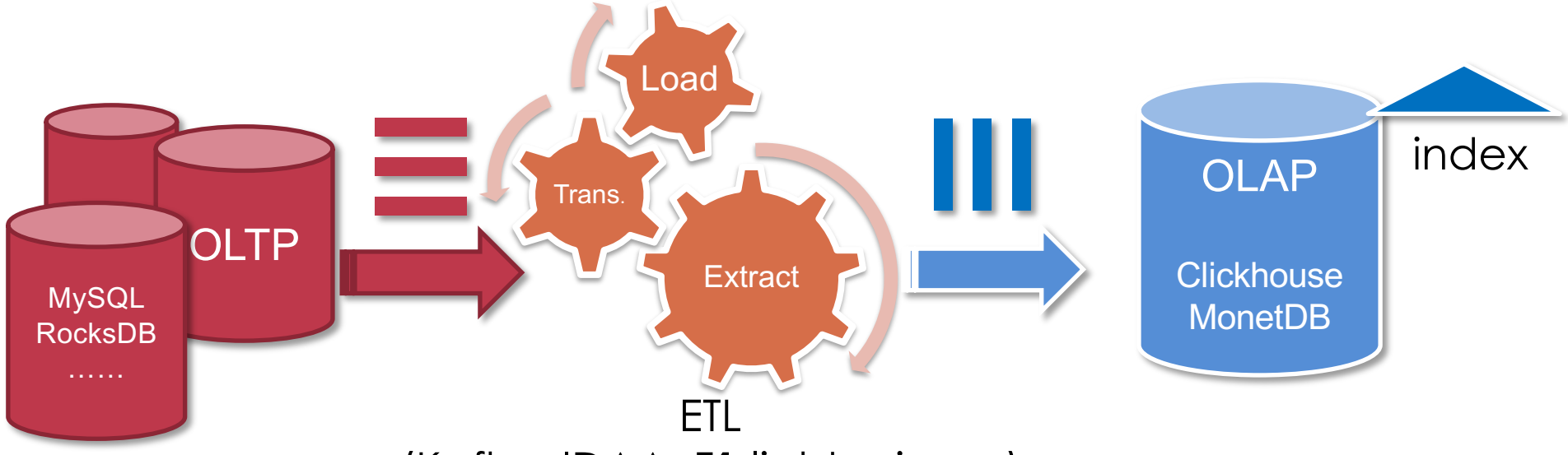
Freshness

- Real-time time delay between TP and AP (e.g., < 20ms)

Scenario	Fraud Detection	System Monitor	Personalized Ad.	Stock exchange
Time delay	**20 ms**	**20 ms**	100 ms	200 ms

[1] ALIBABA CLOUD. Double 11 Real-Time Monitoring System with Time Series Database.

Data analysis: TP+ETL+AP



ETL
(Kafka, IDAA, F1 lightening…)

Data generation, row store
(update, delete, insert)

Data analysis, column store
(SUM, AVG, GROUP)

Alternative#1 : DUAL-SYSTEM
Good performance
Time delay: from seconds to minutes

HTAP alternatives

Alternative#1 : DUAL-SYSTEM

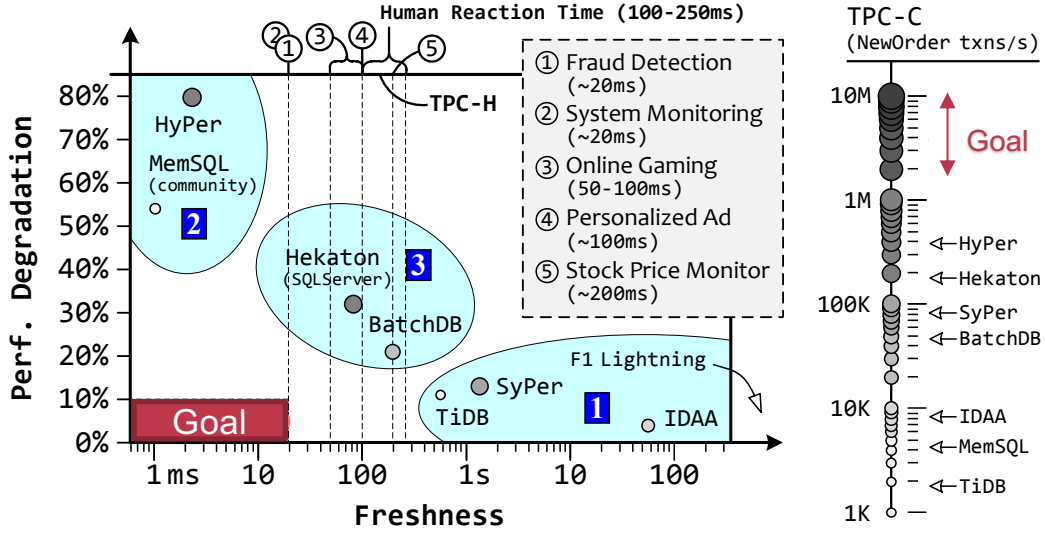
- Good performance
- Large time delay

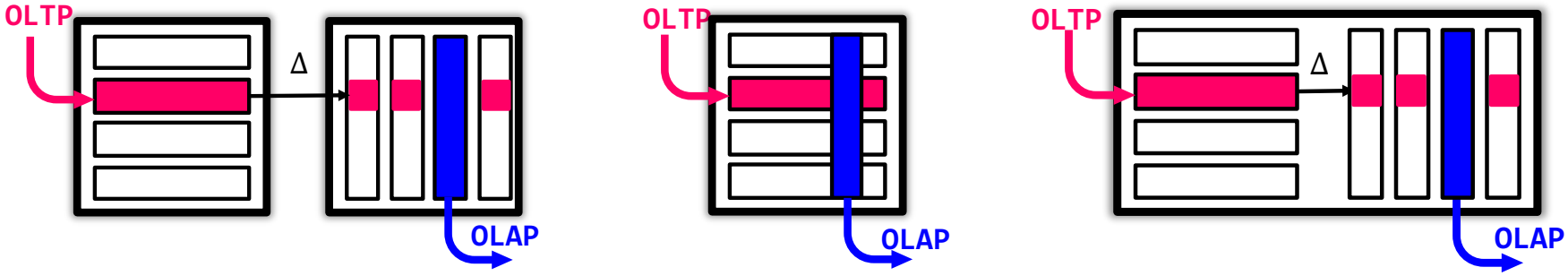
Alternative#2 : SINGLE-LAYOUT

- Short time delay
- Huge perf. degradation

Alternative#3 : DUAL-LAYOUT

- Lightweight sync. (a tradeoff)





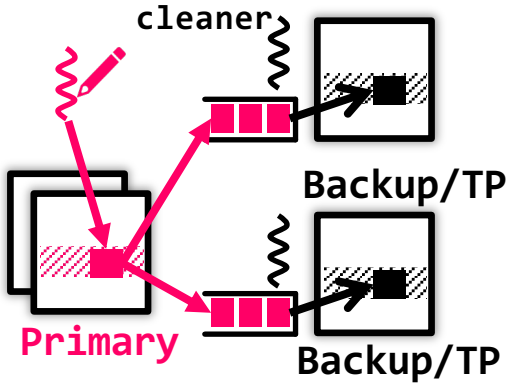
VEGITO
a distributed in-memory HTAP system

Opportunity of High Availability for fast in-memory OLTP

- Replication-based HA mechanism is common
- Synchronous log shipping during transaction committing

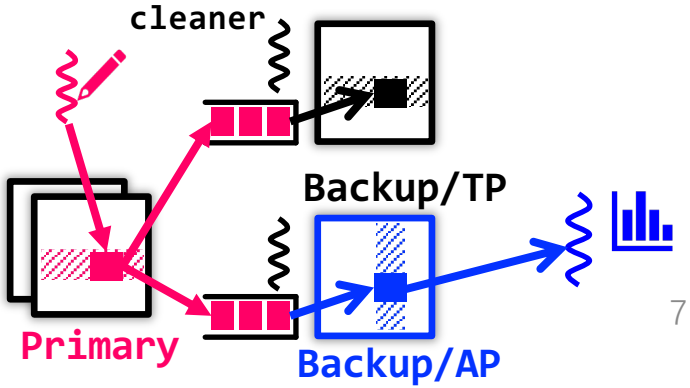
Reuse HA for HTAP

- For performance: OLTP on **primary**, OLAP on **backup/AP**
- For freshness: synchronous logs

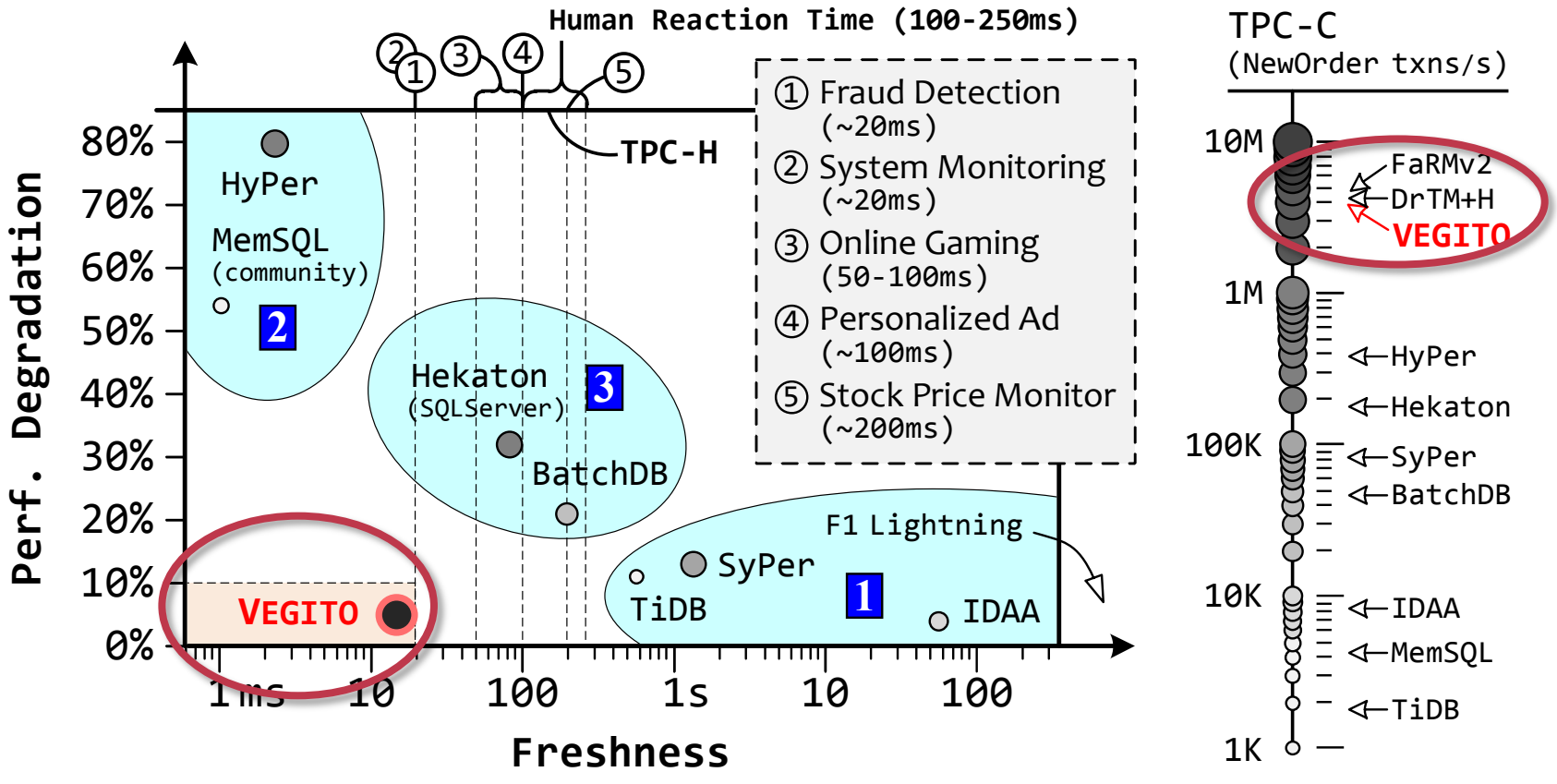


Backup/TP:
- Fresh
- Fault-tolerant

Backup/AP:
- Fresh
- Fault-tolerant
- Columnar



Effects of VEGITO



Goal of Backup/AP: fresh, fault-tolerant, columnar

CHALLENGES AND DESIGNS

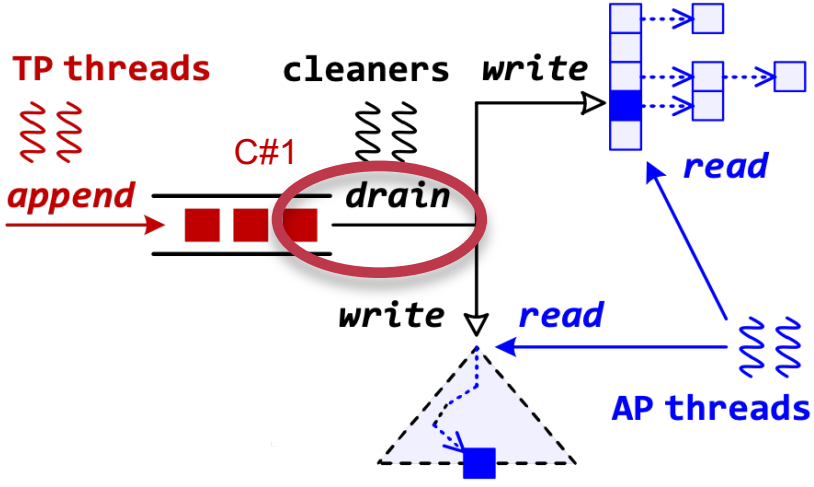
Challenge#1: Log cleaning

Log shipping

- TP threads append logs to queue
- Cleaners drain logs

For high availability

- Drain logs in parallel
- Without consistency until recovery



Backup/AP needs consistent and parallel log cleaning.

For AP queries

- Should be consistency
- Slow (70% ↓): global timestamp + sequential cleaning

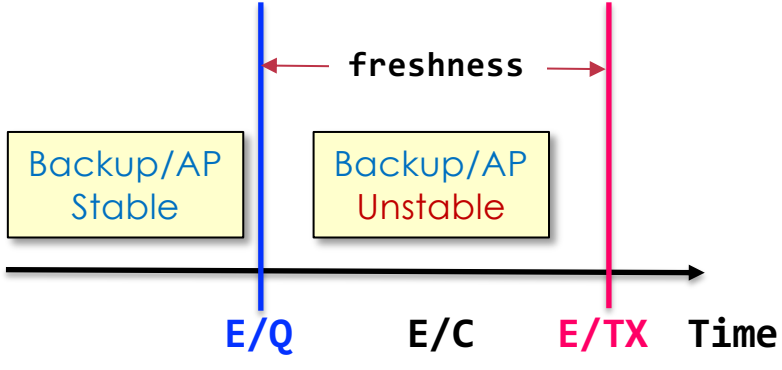
Epoch-based design

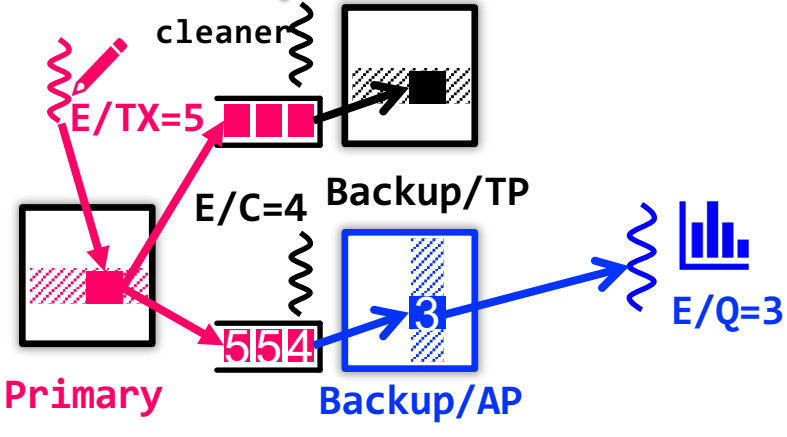
Partition time into non-overlapping epochs

Time isolation between OLTP and OLAP, each machine has

- E/TX: epoch of TX logs (increase periodically)
- E/C: epoch of logs being drained
- E/Q: epoch of stable versions on backup/AP

Freshness: ms-level epoch with consistency of distributed TX

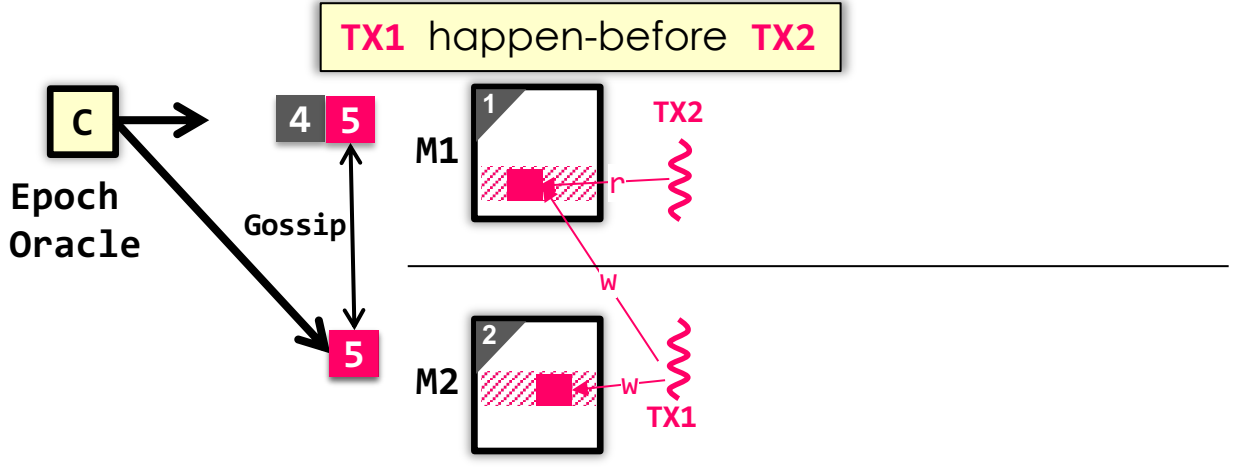




Consistent epoch assign

Gossip-style epoch assign

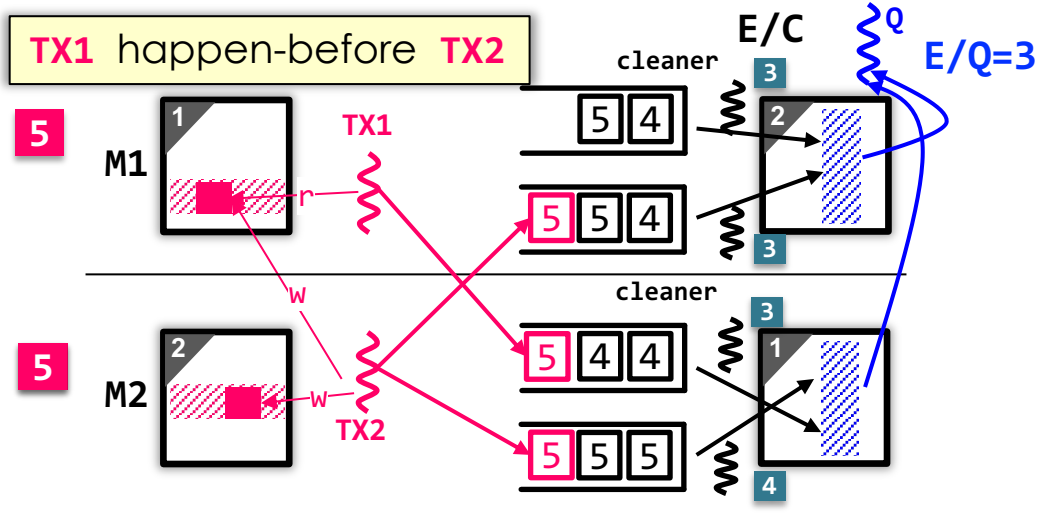
- Epoch oracle: update epoch periodically and broadcast
- Gossip epoch during commit if violate dependence
- Consistency: previous TX within an equal or smaller epoch



Parallel log cleaning

Clean logs matching TX dependence

- Parallelism: Logs within an epoch drained in parallel
- Consistency: each machine update **E/C** when all logs of an epoch drained individually



Challenge#2: MVCS

Multi-version column store (MVCS)

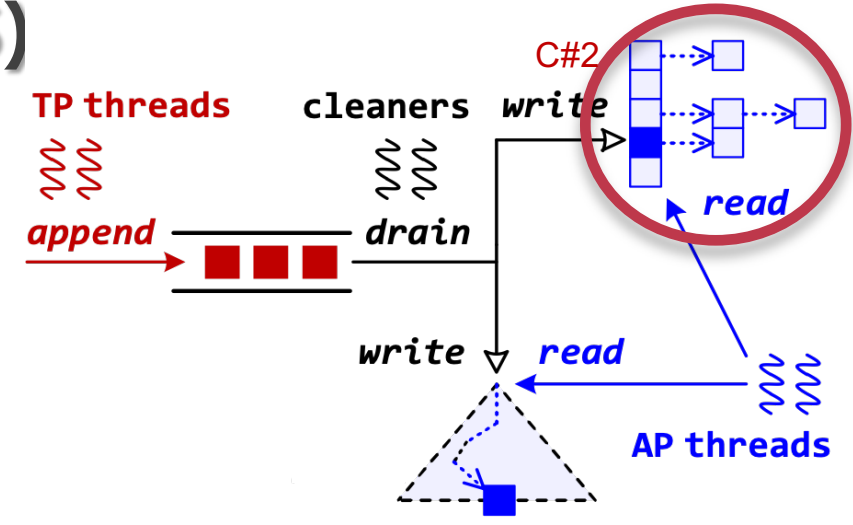
- Isolation & OLAP performance

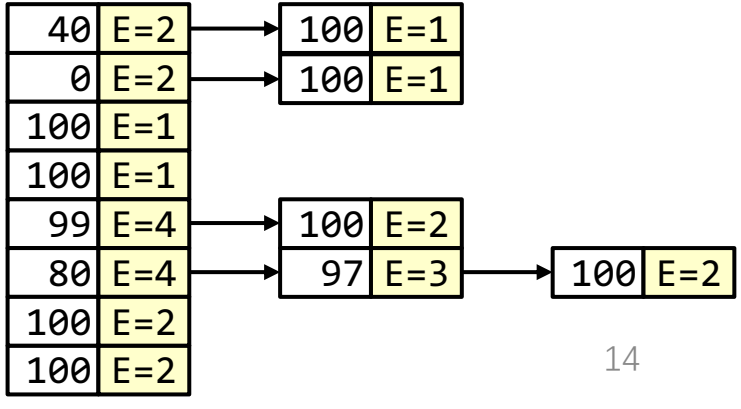
Different locality for read & write

- column-wise vs. row-wise

Chain-based MVCS

- Update efficiently
- Scan performance drop 90%
- when read 0.5 more version on avg.





VEGITO: block-based MVCS

Multi-version column store

Exploit performance for OLAP

- Scan-efficient (locality)
- Update: CoW in the unit of blocks

E=4	E=3	E=2	E=1
40	40	40	100
0	0	0	100
100	100	100	100
100	100	100	100
99	100	100	
80	97	100	
100	100	100	
100	100	100	

Optimizations: Row-split & Col-merge

- temporal & spatial locality
- Split a column into several pages (unit of CoW)
- Merge high-related columns together

12.5x scan performance improved

Challenge#3: tree-based index

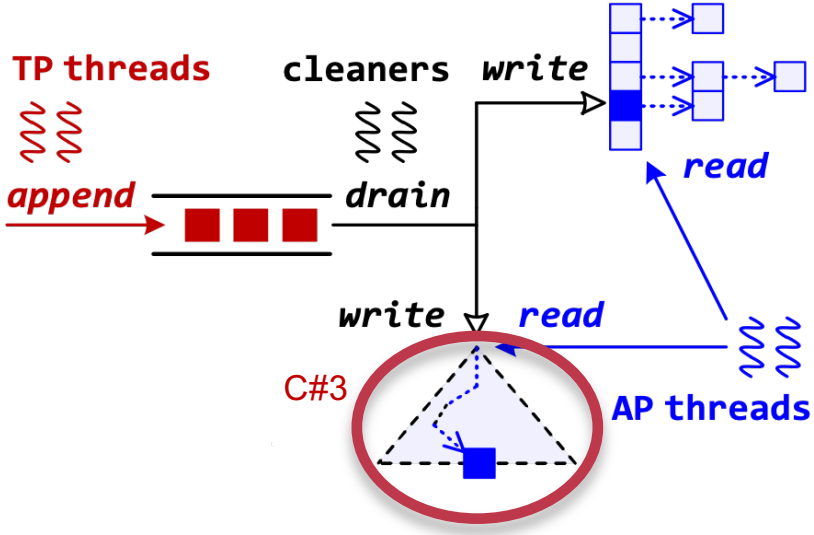
Interference from heavy inserts

Write-optimized tree index

- At the expense of read performance

Read-optimized tree index

- Write performance is limited

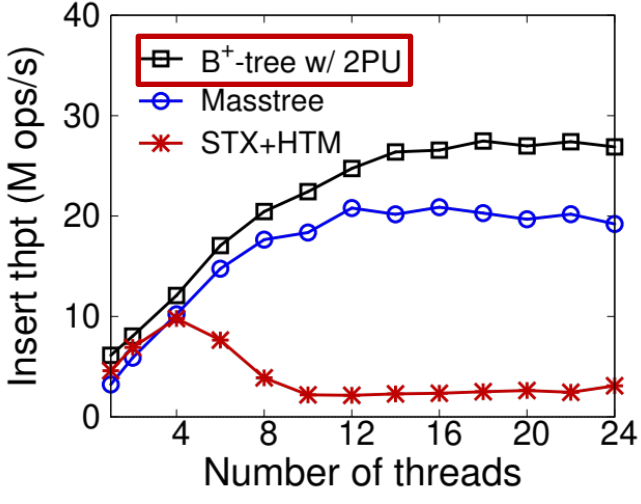


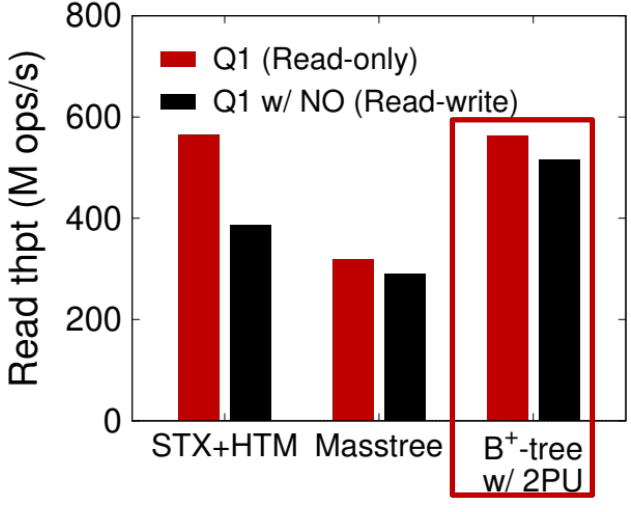
Two-phase concurrent updating

Tree insert = in-place insert + balance (costly)

Insert in buffer of each leaf, balance in batch

- Insert: 8.7x STX+HTM (read-opt), 1.4x Masstree (write-opt)
- Read: 9% overhead under insertion





Fault tolerance

VEGITO perseveres the same availability guarantees

- no need for extra replicas
- prefers to recover the primary from backup/TP

Special cases (rare)

- **Both primary and backup/TP fail**: rebuild primary from backup/AP and migrate to another machine
- **Backup/AP fail**: rebuild to the next epoch

Evaluation Setup

16 machines, each has

- 2x12 Intel Xeon E5-2650 processors,128GB RAM
- 2x ConnectX-4 100Gbps InfiniBand NIC

Benchmark

- CH-BenCHmark ≈ TPC-C + TPC-H

Workload Settings

- OLTP-only
- OLAP-only
- HTAP (VEGITO: epoch interval = 15 ms)

Compare to specific systems

Compared with OLTP-specific systems

- Peak throughput: 3.7 M txns/s
- 1% lower than DrTM+H (OLTP-specific)

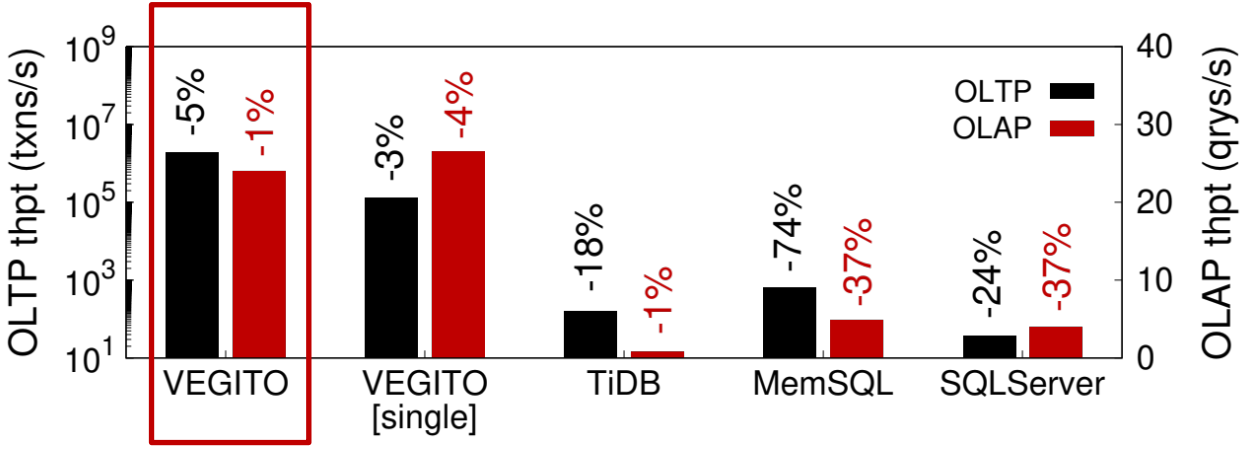
Compared with OLAP-specific systems

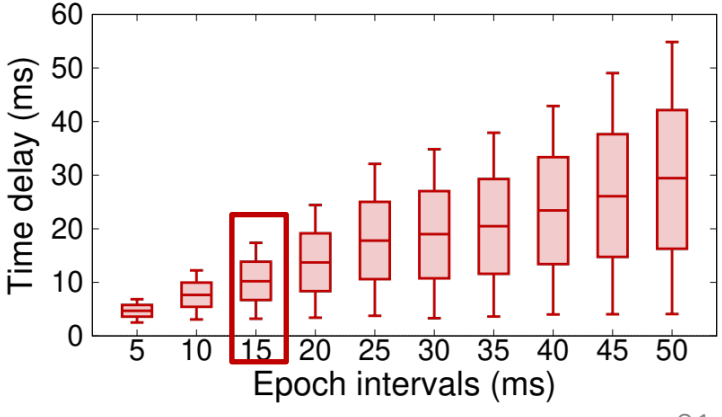
- Geo-mean latency: 57.2 ms
- 2.8x faster than MonetDB (OLAP-specific)

HTAP workloads

VEGITO: performance & freshness

- OLTP 1.9M txns/s (degradation: 5%)
- OLAP 24 qry/s (degradation: 1%)
- Freshness (max time delay) < 20 ms



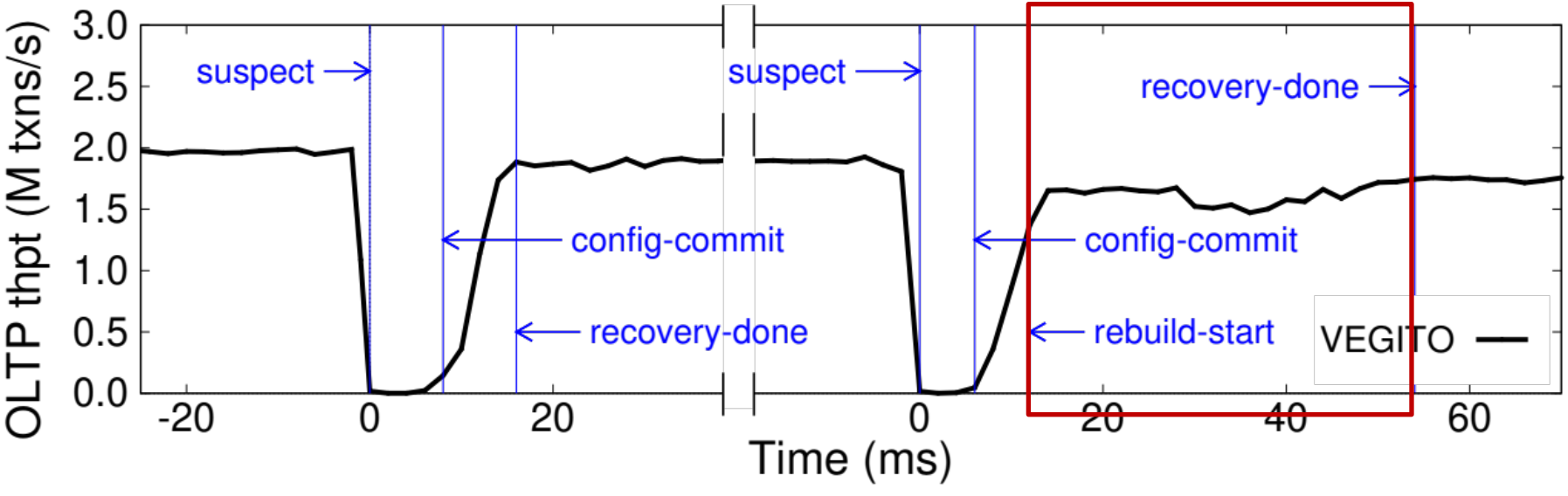


Recovery

Kill one of the primary for twice

Recovery from Backup/AP

- 42 ms for rebuilding the primary



Conclusion

VEGITO : retrofitting high availability mechanism to tame hybrid transaction/analytical processing

OLTP on primary, OLAP on backup

- Backup/AP: **fresh**, **columnar**, and **fault-tolerant**

Please check VEGITO at

- https://github.com/SJTU-IPADS/vegito

Thanks & QA