Tencent Cloud

TDSQL-Boundless:
A Distributed Database System for Large-scale Heterogeneous Multi-Table Workloads

Yuxing Chen, Jie Jiang, Xiaolong He, Ziyi Lu, Shuo Han, Anqun Pan, Chong Zhu, Chang Liu, Xuan Zhao, Yu Li, Yan Tang, Dongzhi Zhao, Wen Zhang, Hailin Lei, Lixiong Zheng

Tencent Inc.

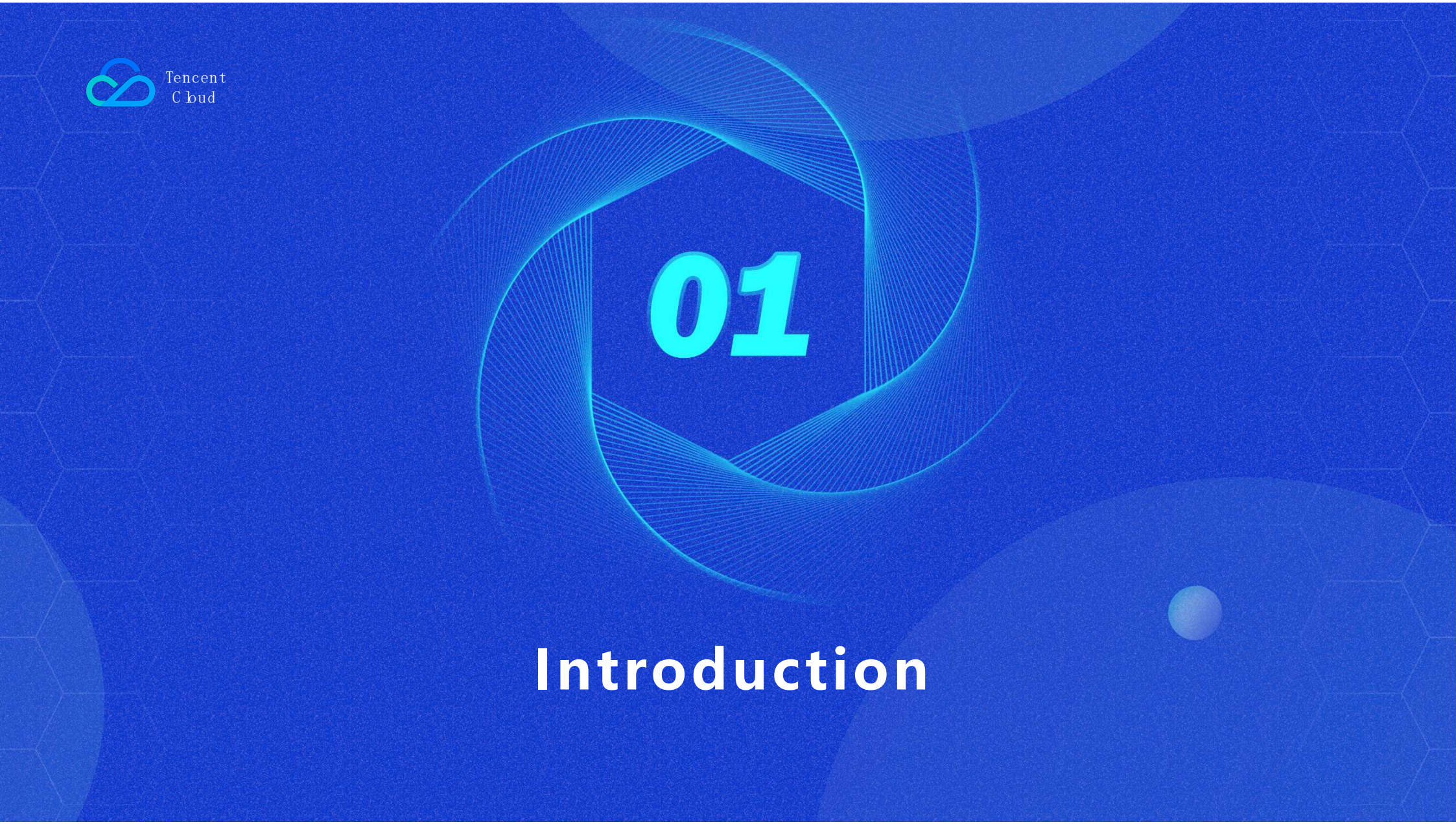
Feng Zhang, Mingde Zhang, Yiteng Chu, Wei Lu, Yunpeng Chai, Xiaoyong Du.

Renmin University of China

axingguchen@tencent.com | Senior Engineer at Tencent Cloud Databases

Tencent Cloud

Content


Tencent
Cloud
01
Introduction

Tencent: Who we are?

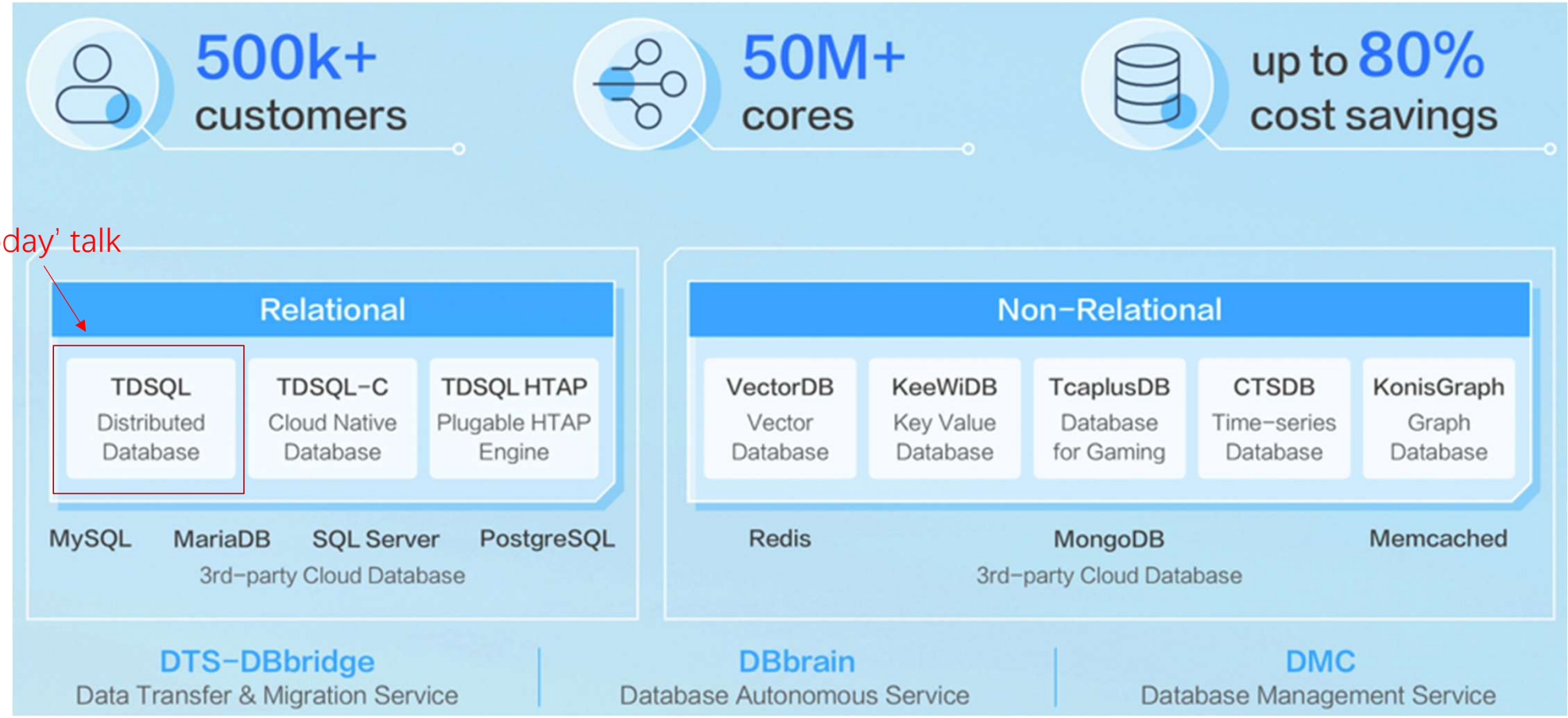
Social media	• 1.3 billion monthly active users
Gaming	• Honor of Kings, League of Legends
Investment	• Epic Games, Spotify, JD.com, PDD

TDSQL: Who we are?

Tencent Distributed SQL

Phase 1 from 2007	• **Internal:** TDSQL was created when Tencent experienced explosive business growth
Phase 2 from 2009	• **External:** Tencent launched its open platform, and TDSQL expanded its services beyond internal usage to cater to the industry internet
Phase 3 from 2014	• **Financial:** TDSQL entered the third stage by targeting prominent financial clients
Phase 4 from 2022	• **Scaling:** TDSQL aims to reproduce the current technology to scale to various industries

TDSQL Product Matrix



Tencent Cloud

02

Applications

Scale and User Cases

Financial

Active users: 1.3 billion

Daily transaction: 1 billion

(0.58 billion from Visa Global 2023)

Entertain ment

Active users: 100 million

Daily transaction: 10 million

(The player count is much higher)

1,000,000 +

Tencent Cloud Database
Core Number

100 + PB

Tencent Cloud Database
Billing Storage

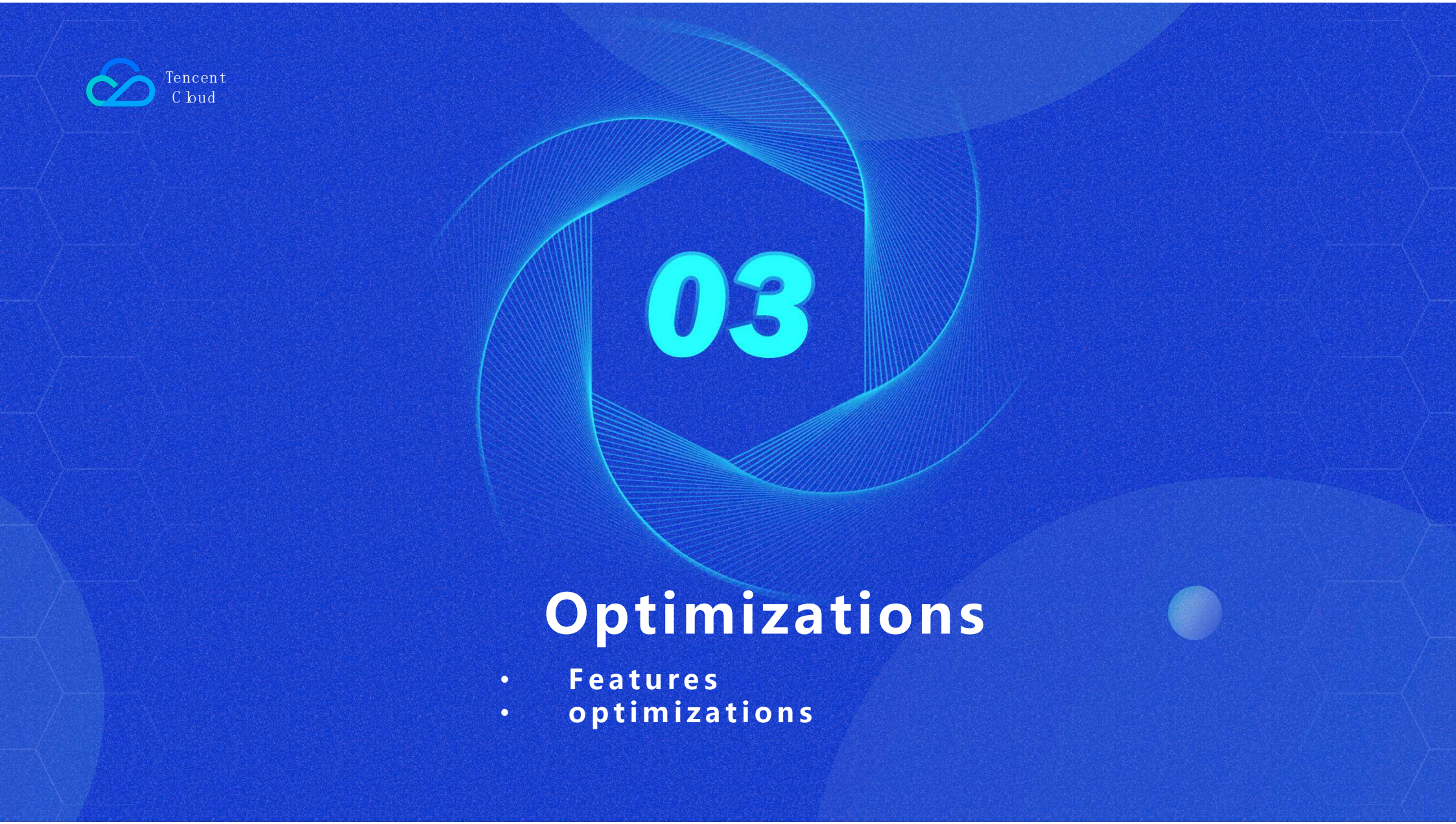
Typical Workloads for This Paper

Tencent

6,000+	500K+	10TB+	50%
Tables per cluster in real deployments	Writes per second at peak load	Metadata for 100K+ regions	Storage wasted by cross-table overhead

Key observations:

- Table count grows unbounded as microservices decompose monoliths
- Each table needs its own partitioning strategy, indices, and lifecycle
- Cross-table operations (JOINs, foreign keys) become prohibitively expensive
- Metadata management becomes the bottleneck, not data processing


Tencent
Cloud
03
Optimizations
Features
optimizations

System Evolution

Shared Disk | Oracle RAC era

Single DB, multiple nodes share storage

- Scalability limited by shared storage
- Cache coherence overhead
- Single point of contention

Shared Nothing | MySQL Sharding era (TDSQL (Our traditional architecture))

Each node owns its data subset

- Manual shard management nightmare
- Cross-shard JOINs expensive
- Rebalancing ~= downtime

Compute-Storage Separation | Current generation (TiDB, CockroachDB)

Compute & storage scale independently

- Better scalability achieved
- BUT: metadata scales with tables
- Transaction cost still high even single node
- Storage inefficiency remains

Our solution: TDSQL-Boundless

SQLEngine (Compute)

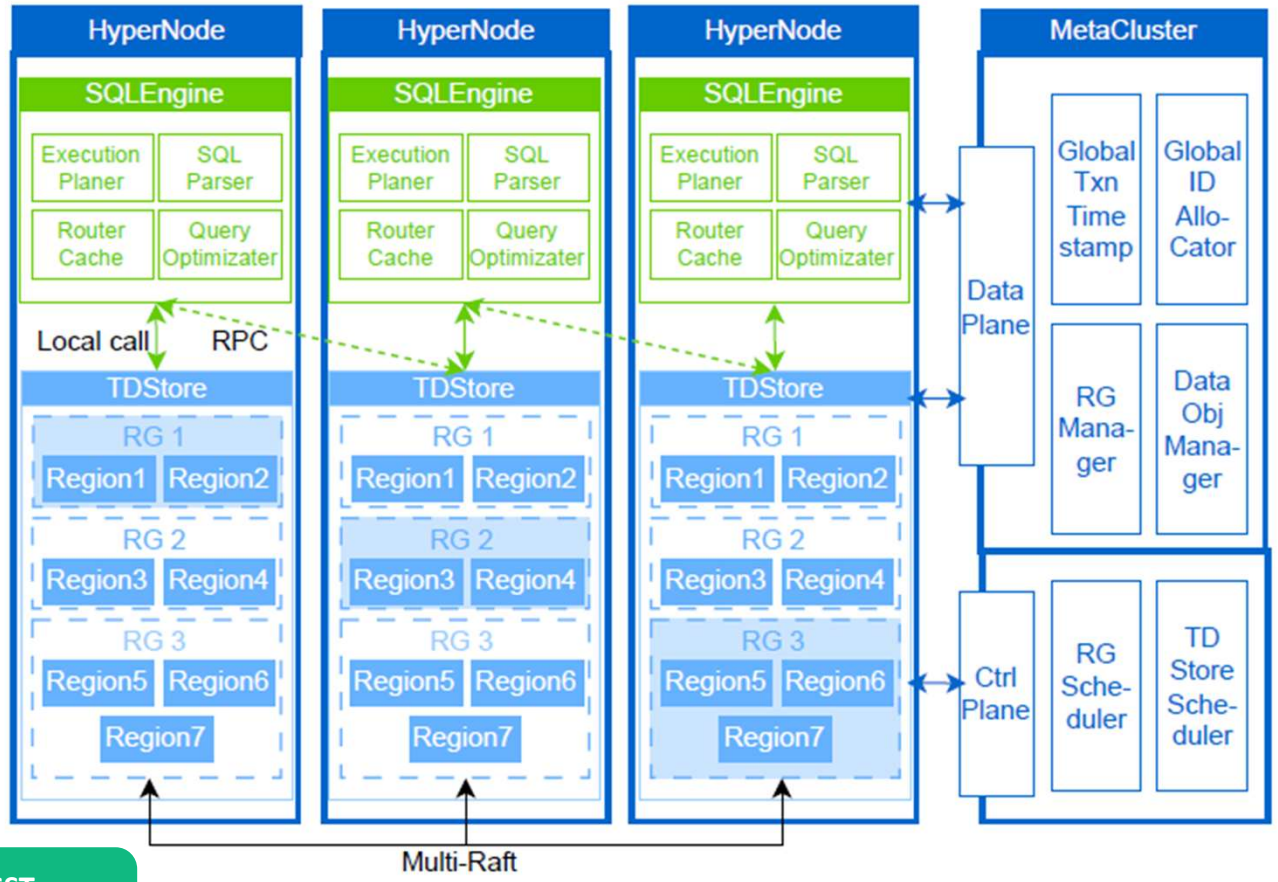
- Stateless SQL parsing & optimization
- Distributed transaction coordination
- Sends requests to TDStore

TDStore (Storage)

- Multi-Raft consistency (Innovation #1)
- Three-tier: DataObject->Region->RG
- SST Boundary Alignment (Innovation #2)
- LSM-tree based KV storage

MetaCluster (Management)

- Cluster metadata management (etcd-based)
- Unified timestamp oracle
- Intelligent load balancing
- One-primary-multi-secondary deployment



Innovation #1: Multi-Table Raft Group

Innovation #2: SST Boundary Alignment

Three Tier Storage Model

DataObject — Smallest unit

1 SST file per table-partition

- Logical container for single (table, partition)
- Maps to exactly 1 SST file on disk
- Independent lifecycle per table
- ~10KB memory footprint each

Region — Replication unit

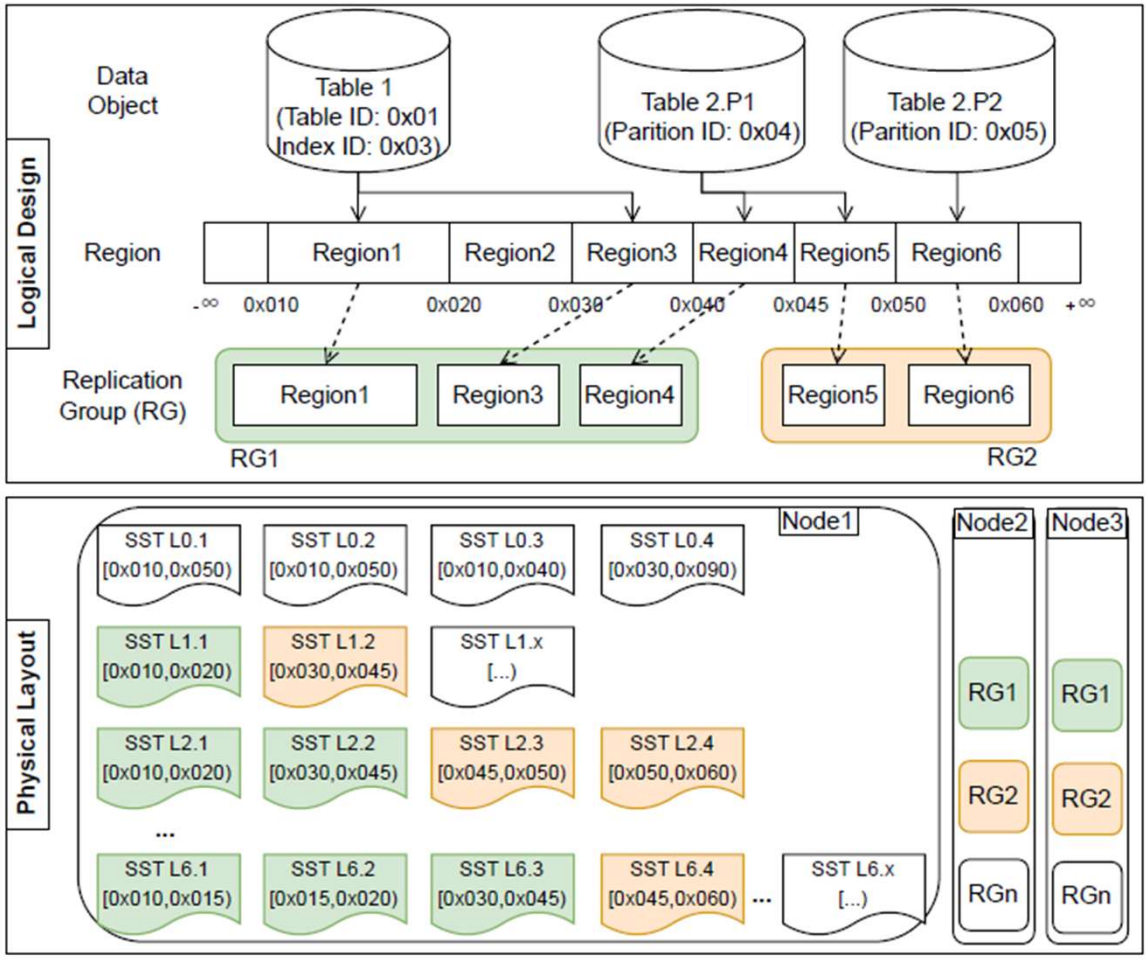
Set of DataObjects forming Raft group

- Multiple DataObjects form one Raft group
- Shared consensus across tables
- Reduces Raft groups from N×K to K
- Metadata reduced by 10-100x

RegionGroup (RG) — Scheduling

Batch of Regions for placement

- Co-located for locality
- Atomic migration/split/merge
- Load balancing granularity
- Enables bulk operations



Innovation #1 Multi-Table LSM-tree Architecture

Traditional approach

Each table = separate LSM tree

N tables × K levels = N×K compaction tasks

TDSQL-B approach

One shared LSM tree, tagged SST files

N tables share K levels = K compactions total

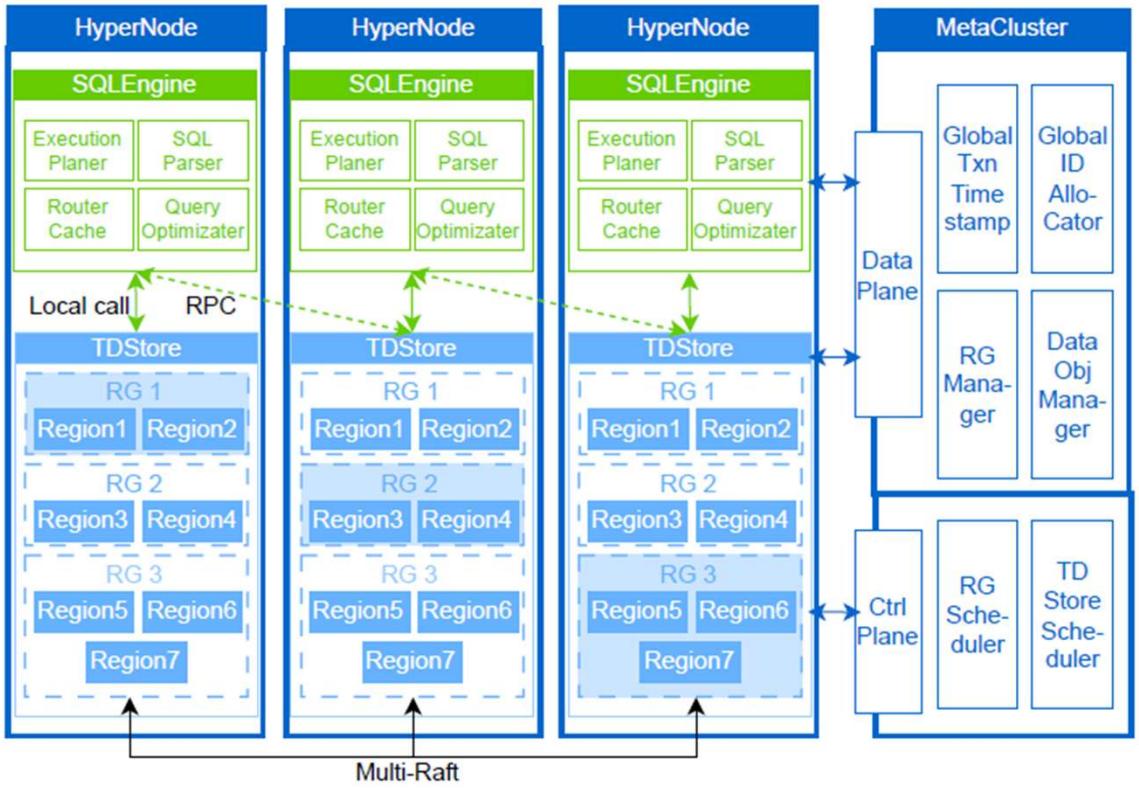
Benefits:

✓ Compaction I/O amortized across all tables

✓ L0 bloom filter shared (no per-table overhead)

✓ Write amplification reduced significantly

✓ Small tables don't trigger unnecessary major compactions



KEY RESULT: 6000 tables × 100 partitions = 100 RegionGroups (not 600,000 individual Raft groups!)

Innovation #2 SST File Boundary Alignment

Core Idea

SST file boundaries are aligned so that each SST contains data from ONLY ONE table. No cross-table mixing within an SST.

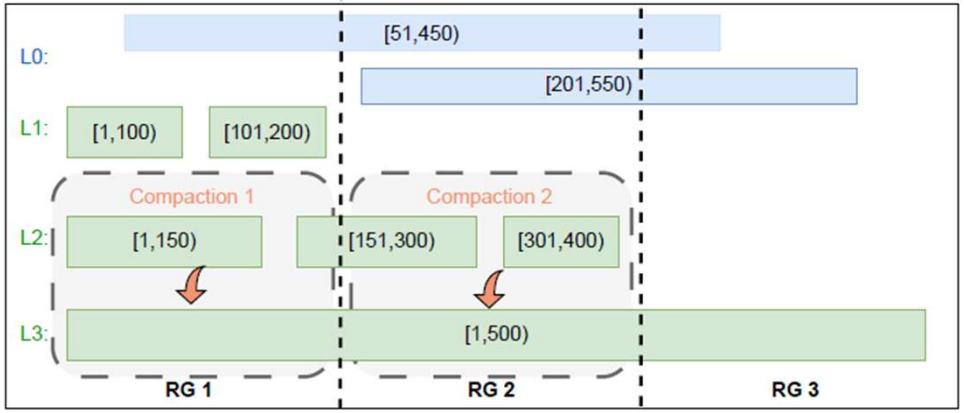
BEFORE

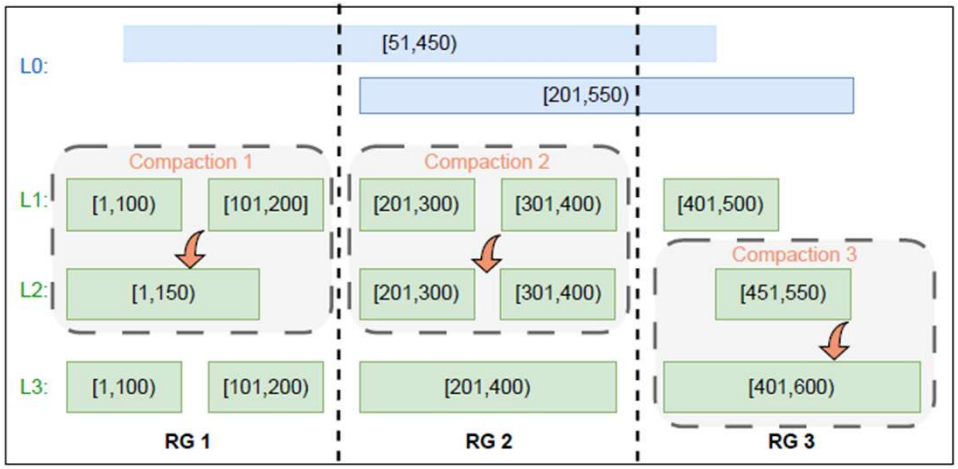
SST contains mixed data from multiple tables -> Read must scan & filter irrelevant data -> Bloom filter can't eliminate false positives

AFTER (TDSQL-B)

SST aligned to single table boundary -> Point query reads exactly 1 SST -> Range scan skips non-matching SSTs entirely

Result: Up to 10x reduction in read I/O for multi-table workloads, especially when hot and cold tables are co-located.





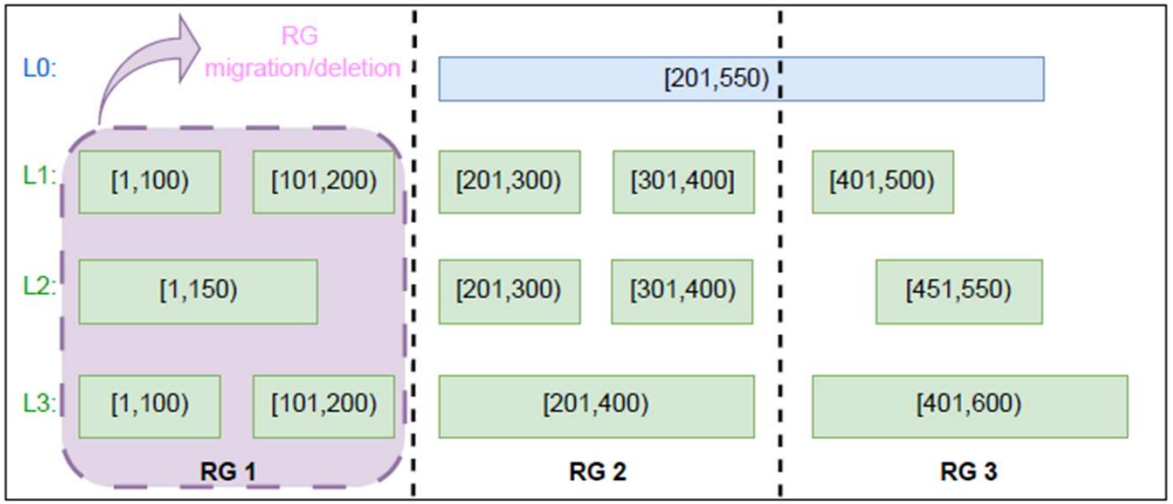
Efficient SST-Level Migration

Problem with Traditional Migration

✗ Migration triggers full compaction of source SSTs
✗ All tables in the region must be compacted together
✗ Migration time proportional to TOTAL region size
✗ Blocks writes during compaction (STW)

TDSQL-B: SST-Level Copy-on-Write

✓ Track SST ownership via manifest (not physical layout)
✓ Migration = update pointers, copy only changed SSTs
✓ Unchanged SSTs shared between source and destination
✓ Zero-downtime: background copy + atomic switch



Measured Improvement

Migration latency: 85% reduction
Data copied: Only delta (avg 12% of region size)

Evaluation Methods

How we measure improvement?

Cluster Configuration

- 3 servers (compute + storage mixed)
- Each server: 375GB RAM, 12x3.5 TB SSD
- Intel(R) Xeon(R) Platinum 8255C 96 cores
- Tencent tLinux release 4.14

Workload Characteristics

- SysBench variant: 6,000 tables (scaled up)
- Benchmark: TPC-C e-commerce scenarios
- Finance: 100 GB daily writes
- Games' Log Stream service: 500k data records write per second

Comparison Baselines

- TDSQL (shared-nothing storage)
- TiDB v7.6.0 (per-range Raft)
- HBase (NoSQL, manual region splitting)
- TokuDB (Storage-compressed baseline)

Metrics Measured

Throughput:	ops/sec at varying concurrency levels
Latency:	p50, p99, max at steady state
Memory Usage:	metadata memory vs table count scaling
Storage Amplification:	Actual disk usage / logical data size
Migration Time:	Time to migrate 1TB RegionGroup

Evaluation – Baseline System

2.0x
Throughput
vs TDSQL

2.0x
Throughput
vs TiDB

3.0x
Throughput
scale to 1024 threads

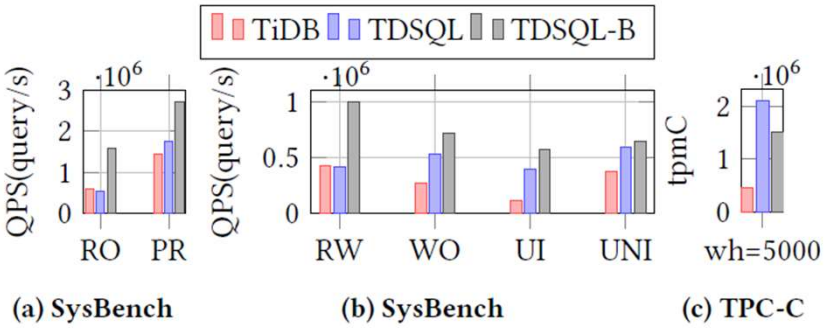
-90%
Memory usage
@ 8K tables

Scaling Behavior

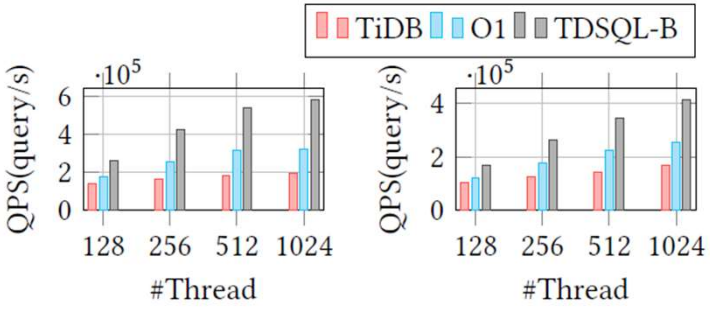
✓ Throughput improvement vs baselines
✓ Metadata memory stays flat (not O(table_count))
✓ Throughput improvement scales 1024 threads
✓ Compaction overhead decreases with more tables (amortized)

Key Insight

Unlike traditional systems where performance DEGRADES with more tables,
TDSQL-B improves efficiency as table count grows due to better amortization of compaction, Raft, and metadata overhead.



(a) SysBench (b) SysBench (c) TPC-C



(a) Read-write(RW) (b) Write-only (WO)

Evaluation – Ablation Study

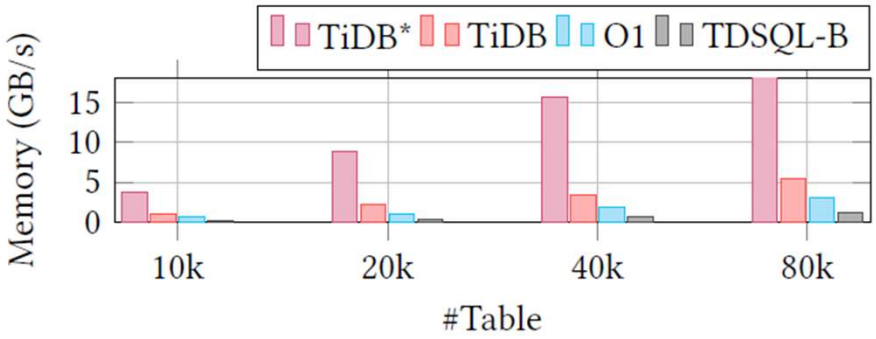
Ablation: Feature Contribution

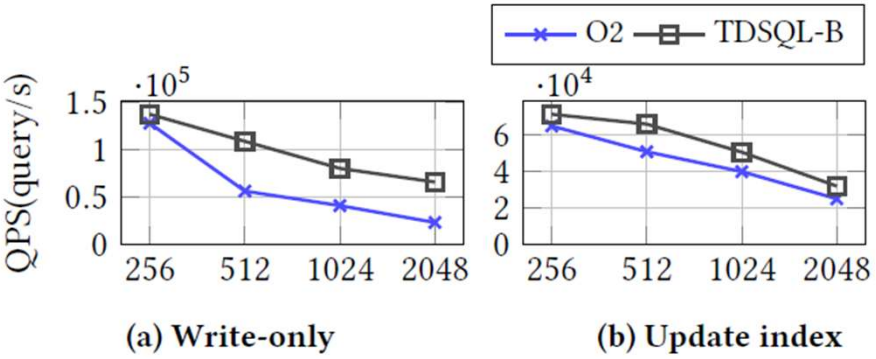
Baseline (no innovations)	1.0x
+ O1:Multi-Table Raft only	~1.65x
+ O2:SST Alignment only	~1.37x
+ Both innovations	~2.26x
+ Physical Migration.	~10x

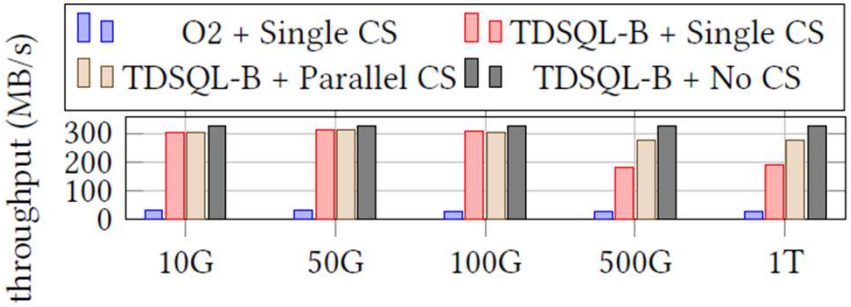
Takeaway

Both innovations are NECESSARY for maximum impact:
- Multi-Table Raft reduces coordination overhead (~65% gain)
- SST Alignment eliminates read waste (~37% gain)
- Combined effect is SUPERADDITIVE (over 2x)

They address different bottlenecks that compound.







Evaluation – Production Impact

Cloud Testing

A Competitor vs. TDSQL-B

- Sysbench
- Query speed: 70% faster for write-only
- Cloud cost: 20% of the cost

Financial Top-up Service

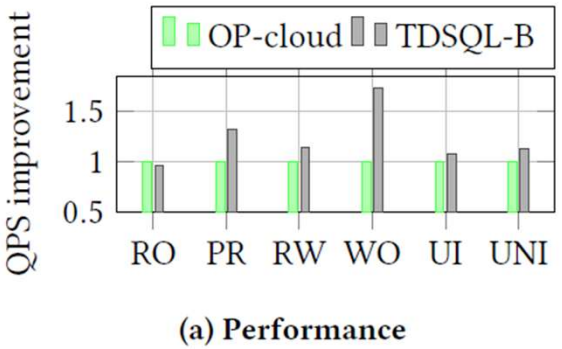
HBase → TDSQL-B

- 100GB data daily growth
- Storage: 35.3TB → 16.6TB (-51%)
- p99 latency: 150ms → 37ms (-76%)

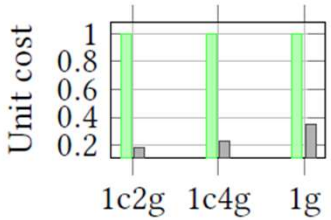
Tencent Game Log Streaming

TokuDB → TDSQL-B

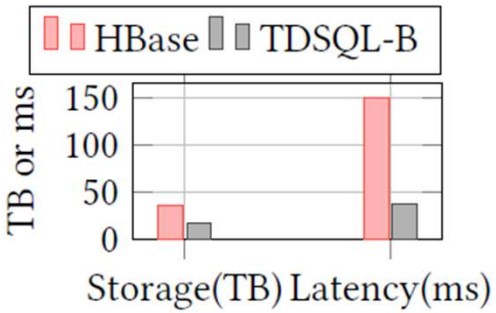
- high write throughput
- Max Writes: 80K/sec → 164K/sec (+100%)
- Nodes: 76 → 23 (-69%)

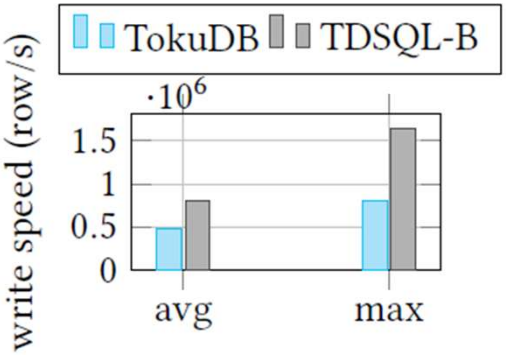


(a) Performance



(b) Cost





Average 50% storage reduction, 70% latency improvement, 60% infrastructure cost savings.

Tencent Cloud

04

Future Work

Research topics

TDSQL
VLDB 2024

TDSQL: Tencent Distributed Database System

TPC-C benchmark

TDSQL-Boundless
SIGMOD 2026

TDSQL-Boundless: A Distributed Database System for Large-scale Heterogeneous Multi-Table Workloads

Distributed-LSM Optimization

TDSQL-HTAP
2027

TODO1: HTAP Design and Optimization

TXSQL: Lock Optimizations Towards High Contented Workloads

2PL Optimization for Hotspot

TXSQL
SIGMOD 2025

Reaching the Pinnacle of TPC-DS: Co-design of Architecture, Executor, and Storage in TDSQL

TPC-DS benchmark

TDSQL
VLDB 2026

TODO2: Object Storage Design and Optimization

TDSQL-COS
2027

Contact us

axingguchen@tencent.com

Qing Yun Plan (High-level Job Position for Graduates)

https://join.qq.com/qingyun.html

WeChat

Thanks

Yuxing Chen
axingguchen@tencent.com

Why Existing Architectures Fall short

Three critical failure modes in production

Metadata Explosion

- Partitions scale with table count
- 20-30GB RAM @ 100K regions
- Risk of Out-of-Memory crashes
- 5-table SQL may route to 50 partitions

~30GB
RAM metadata

Transaction Overhead

- Cross-partition txns require 2PC
- Even same-node cross-region pays cost
- Latency amplification from coordination
- Lock contention across partitions

2x+
Latency increase

Storage Inefficiency

- +40% data bloat (Oracle to MySQL)
- Shared LSM trees cause L0 bloating
- Reduced compaction efficiency
- Small tables suffer heavyweight ops

-40%
Storage loss

These are NOT edge cases - daily reality in banking, gaming, financial services.