Conceptual Modeling on Tencent's Distributed Database Systems

Pan Anqun, Wang Xiaoyu, Li Haixiang

Tencent Inc.

Outline

- Introduction
- System overview of TDSQL
- Conceptual Modeling on TDSQL
- Applications
- Conclusion

Outline

- Introduction
- System overview of TDSQL
- Conceptual Modeling on TDSQL
- Applications
- Conclusion

About Tencent

Social Platform

Digital

QQ

WeChat/WeXin

Video

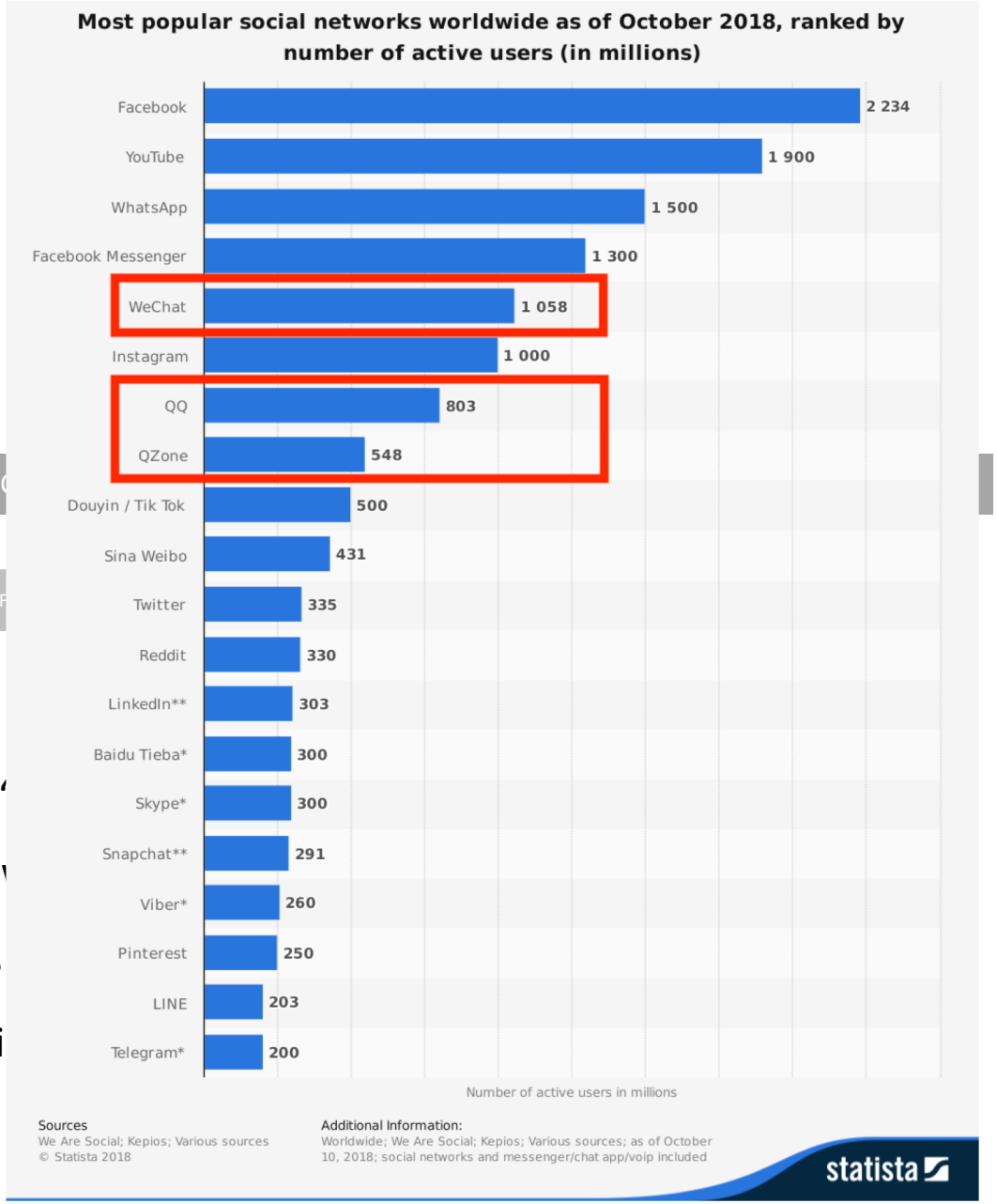
Games

Monthly active user accounts ('

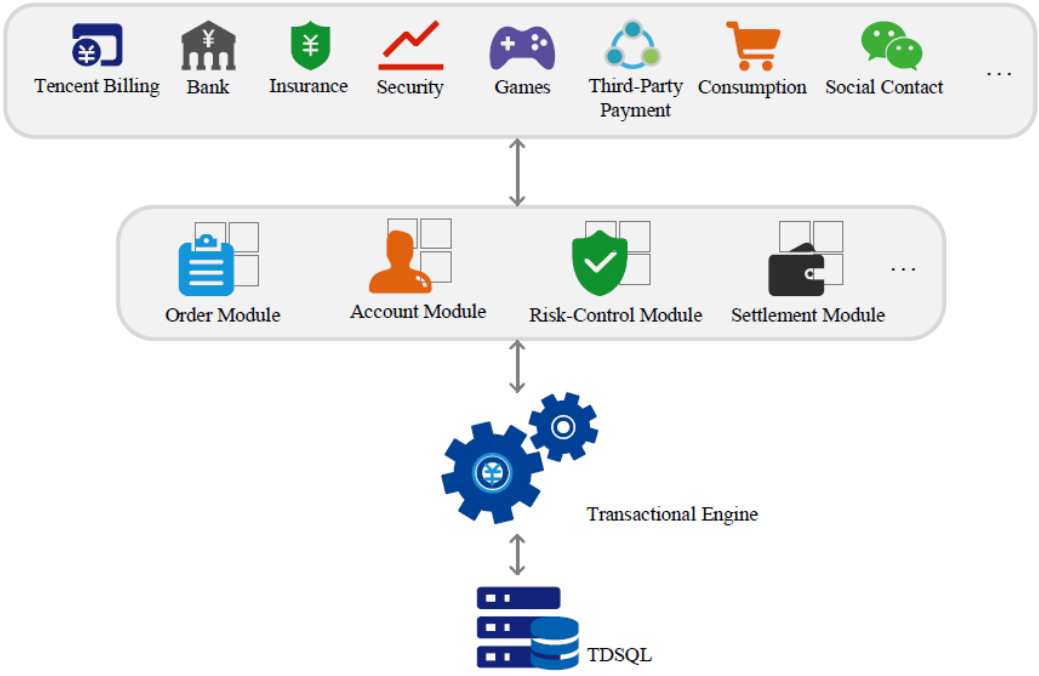
Combined MAU of Weixin and

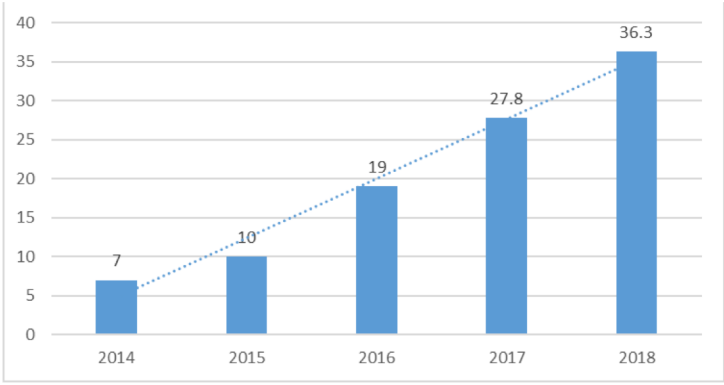
More than **20,000** Applications

More than **1 million** Database i



Tencent Billing System





The number of Digital Account increased by 30%-50% year-on-year

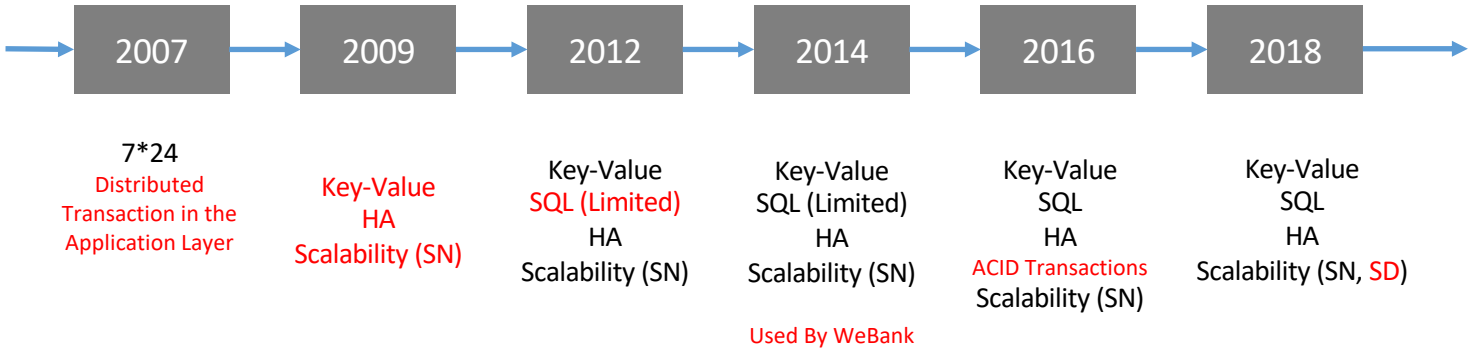
Challenges

- Increasing data volume with growth rate of 30~50% per year
- Dealing with money, ACID, zero data loss;
- Core system, 7*24, more than 800 million Yuan per day
- High cost with more than X0,000 database servers

What database does Tencent need?

- High availability
- Scalability
- ACID Transactions
- Low cost
  - High performance
  - Resource utilization
- Multi-model
  - SQL
  - Key-Value

History of TDSQL



Customers of TDSQL

Outline

- Introduction
- System overview of TDSQL
- Conceptual Modeling on TDSQL
- Applications
- Conclusion

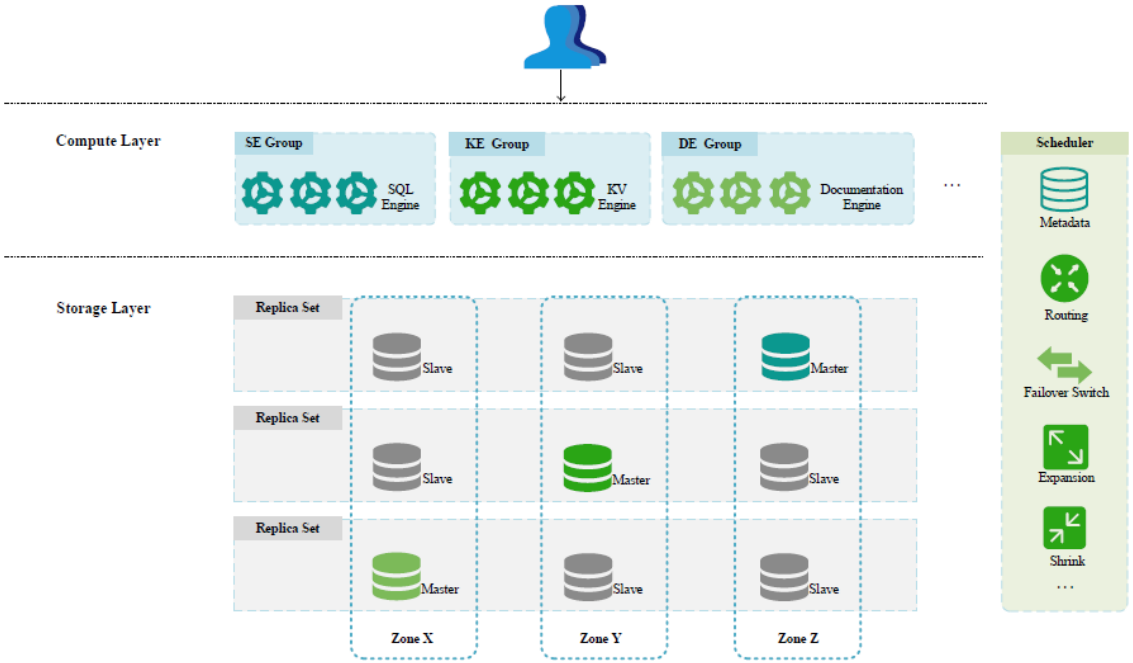
Architecture of TDSQL

Compute Layer: SQL engine (MySQL compatible), KV Engine (such as Redis), Document Engine (such as MongoDB).

In charge of the following operations

1. receiving the request;
2. processing the sql related logic;
3. locating the storage address for storing and computing data through scheduler;
4. exchanging data with storage layer;
5. returning the result.

The engine is Stateless.



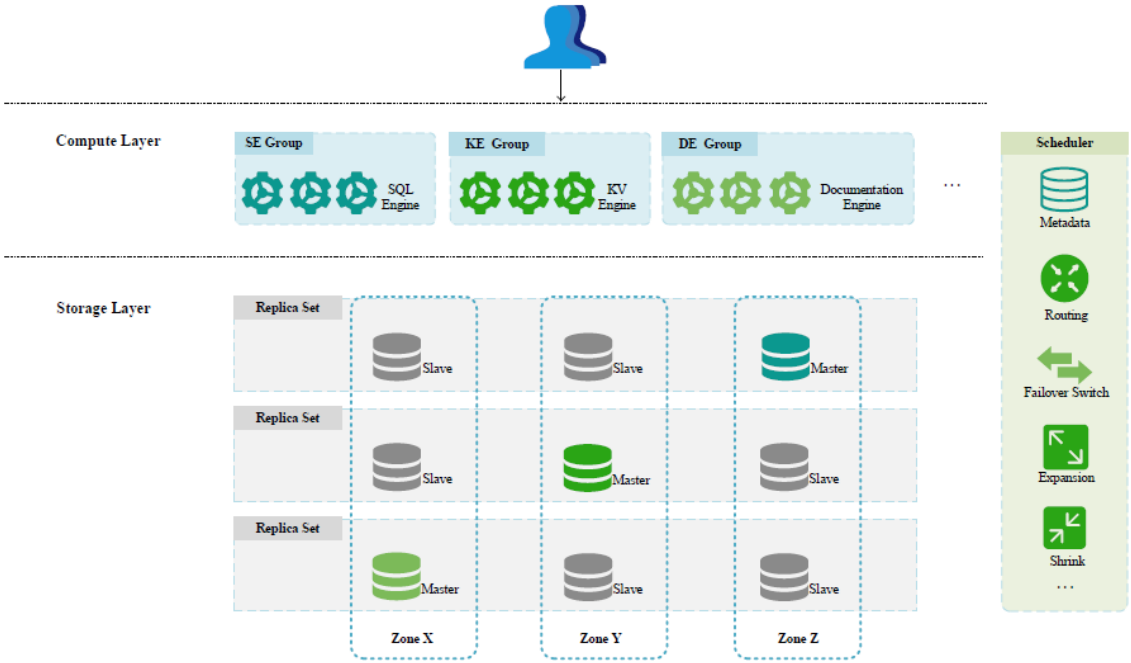
Architecture of TDSQL

Storage Layer: *Data is organized by Replica Set.*

Replica Set: default 3 replicas, Strong consistency replication based on Raft or Asynchronous Replication.

Multiple replica sets can be located in a node.

Table: distributed into multiple replica sets.



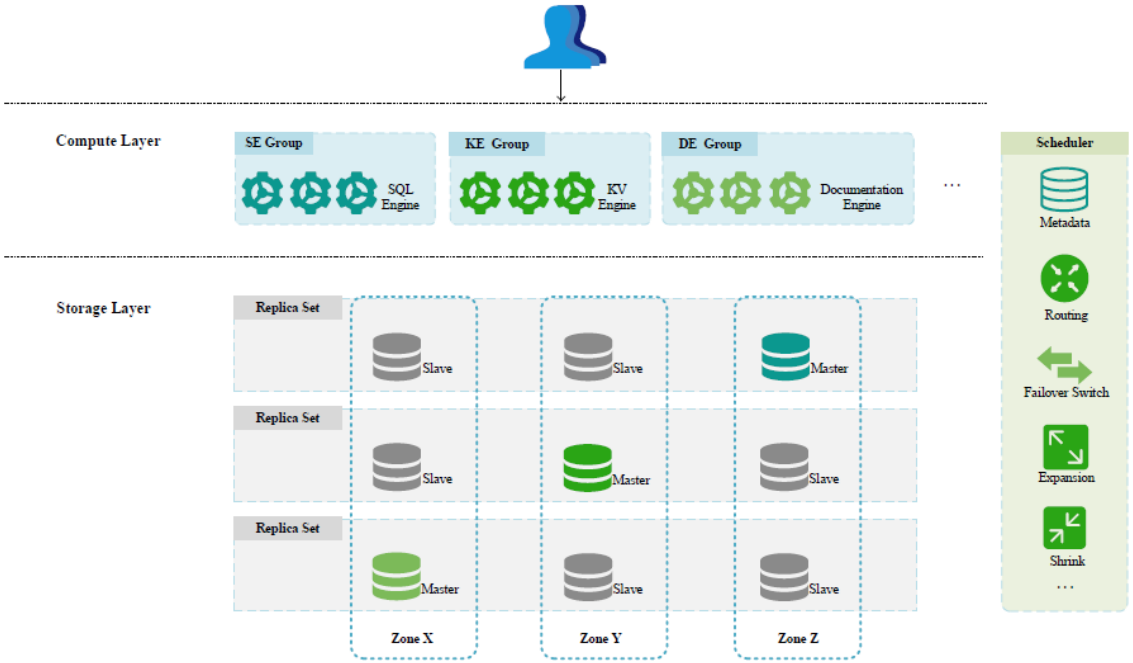
Architecture of TDSQL

Scheduler:

Managing the metadata of the cluster such as the replica set location of a specific key.

Scheduling replica set in the storage cluster, such as data migration, scale out, backup.

Scheduler is a cluster, it needs to be deployed to an odd number of nodes.



Outline

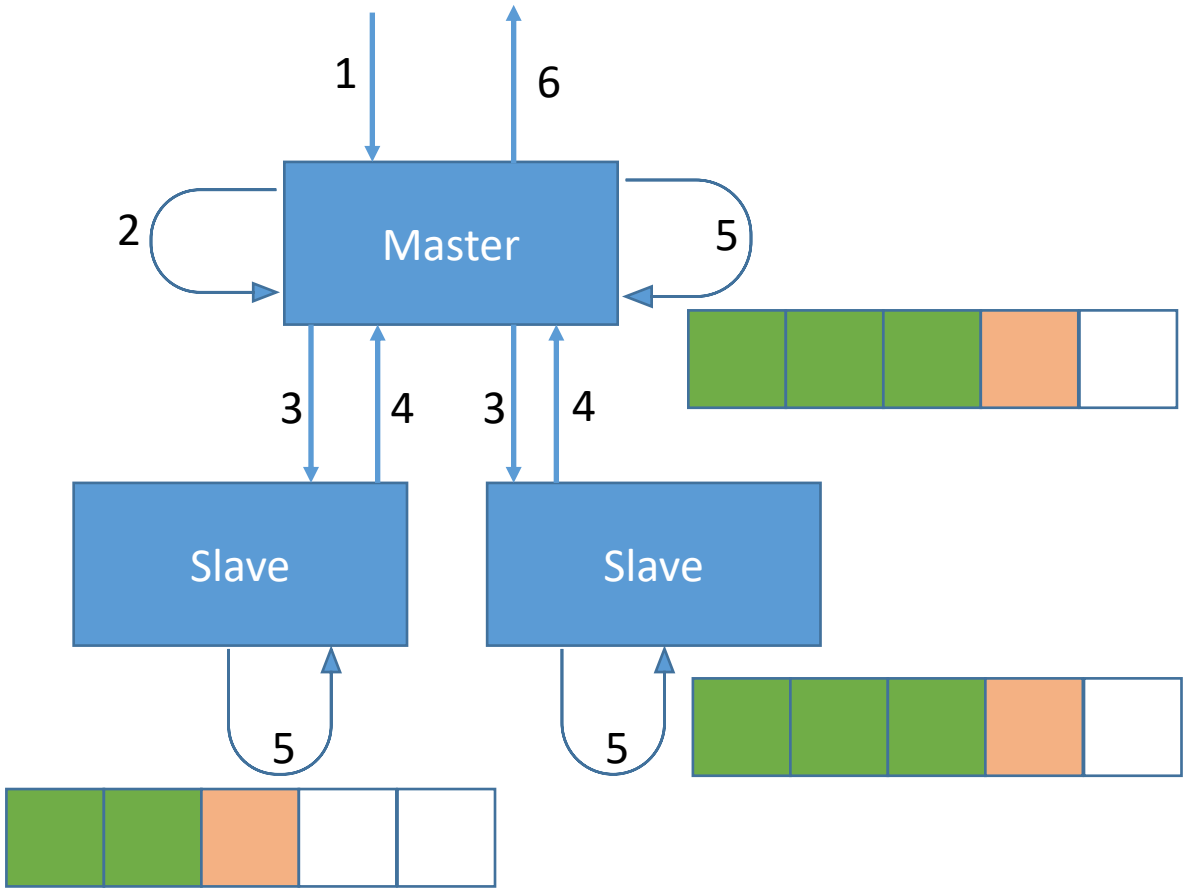
- Introduction
- System overview of TDSQL
- Conceptual Modeling on TDSQL
- Applications
- Conclusion

High Availability

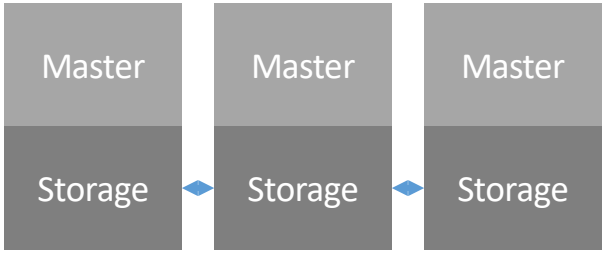
Raft-based Replication

Log Replication：

1. Client send messages;
2. Process messages;
3. Replicate messages;
4. Acknowledge messages;
5. Commit messages;
6. Reply.

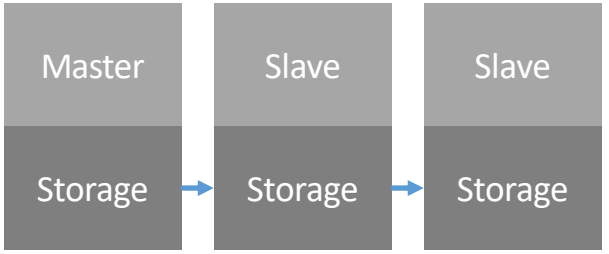


Multi-master or Single-master?



Multi-master Cluster

Weak Consistency
High concurrency
Low latency



Single-master Cluster

Strong Consistency
Higher latency
Lower concurrency

Global Distribution

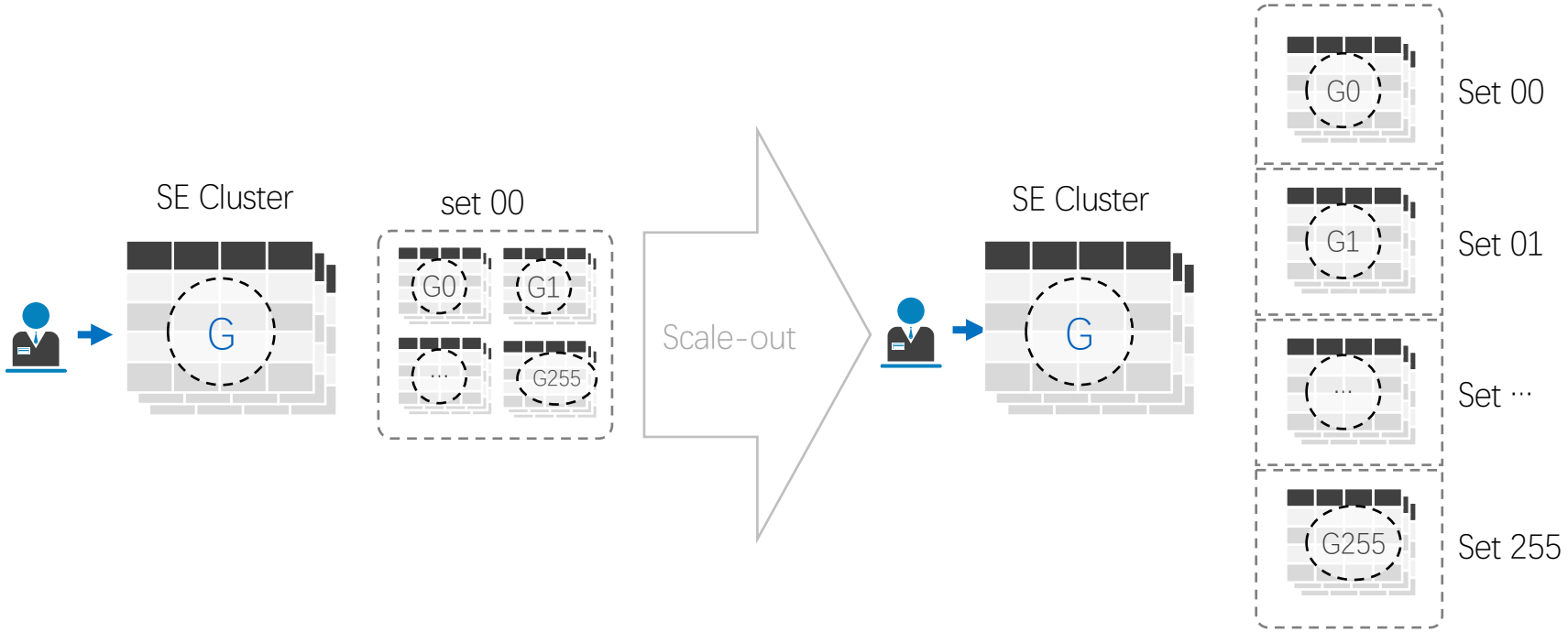
To meet the requirements of customers, we can customize the data distribution policy.

Normally, the replication between zones is strong consistency;

The replication between regions is week consistency.

Horizontal Scalability

Auto-Sharding



1. hash-based partitioning
2. The value of maximum throughput and disk space per set is configured
3. If one set can not server the requirements of application, the set can split into two sets.

Distributed Transactions

Transaction Manager:

1. manage the lifecycle of transactions;
2. stateless, transactions can be routed to any node

Resource managers:

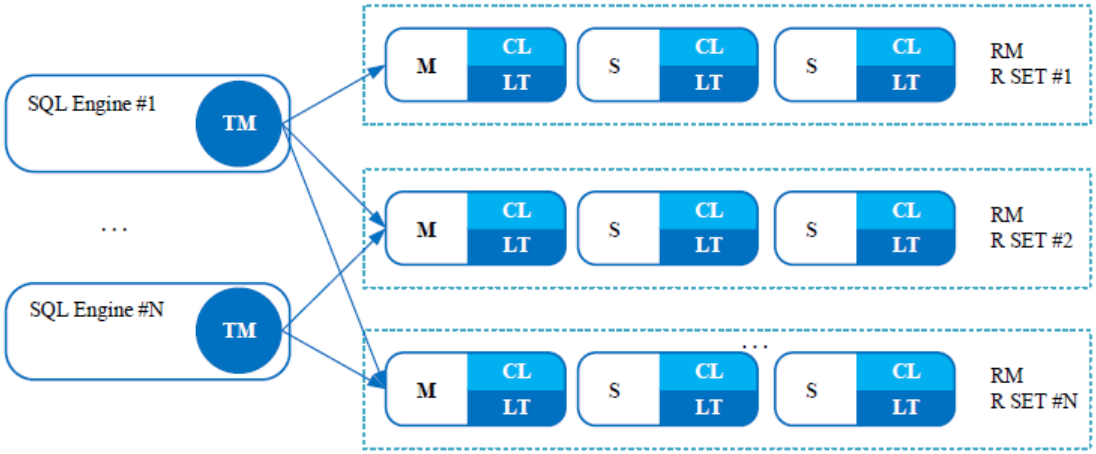
1. execute the queries received from the TM
2. return results to the TM

Commit Log:

1. Global transaction Log

Local Transaction:

1. Transaction involved only the single RM
2. A distributed transaction may include more than one LT

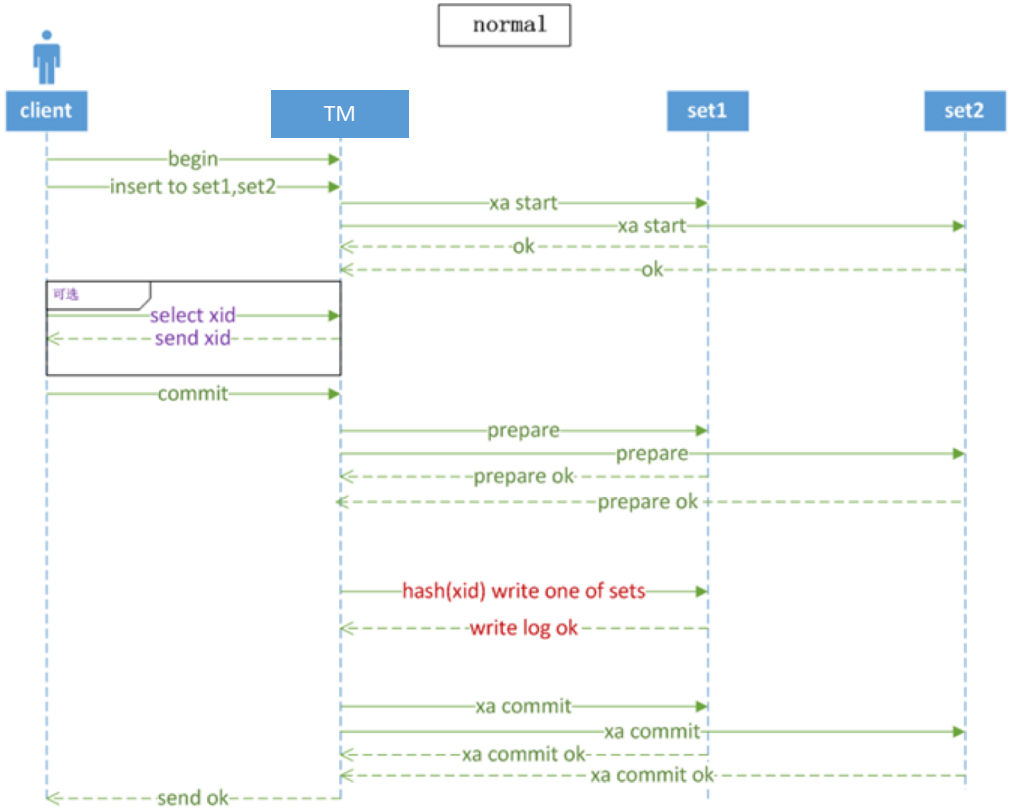


Distributed Transactions

Basic algorithm: 2PC

The following steps outline the lifecycle of a distributed transaction

1. Receive request
2. Assign XID
3. XA start to all set
4. XA prepare to all set
5. Write transaction Commit Log
6. XA Commit to all set
7. Send response



Distributed Transactions

Auto-detect and optimize single-set transactions vs distributed transactions

SINGLE-SET TRANSACTIONS:

A transaction that impacts all the rows only are located in a single replica set.

DISTRIBUTED TRANSACTIONS:

A transaction that impacts a set of rows distributed across replica sets on multiple nodes

Distributed deadlock detection

global wait-for-graphs are created

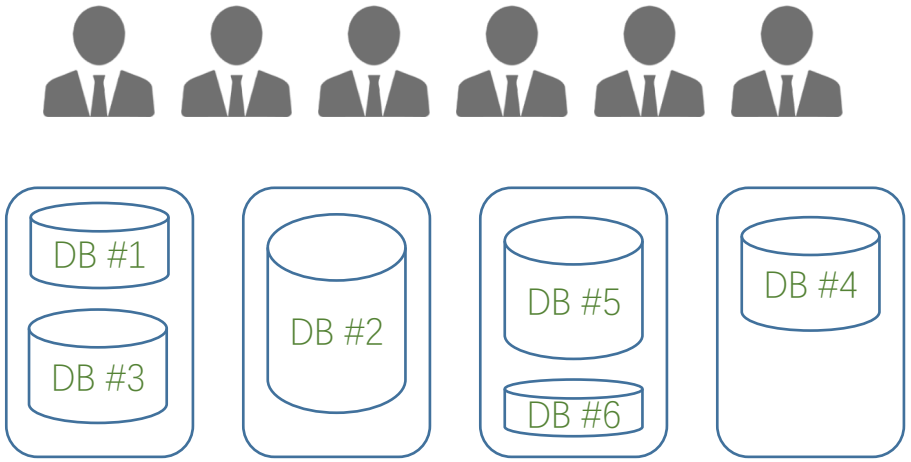
difficult: spot deadlocks since transaction waits for resources across the network

timers: If a transaction does not finish within this time period, the timer goes off, indicating a possible deadlock

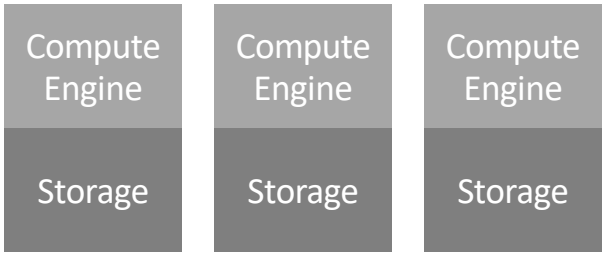
Low-cost

Resource Scheduling

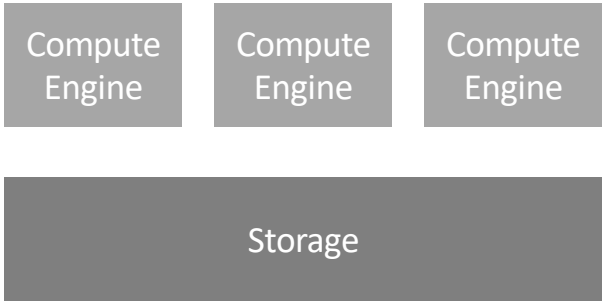
- Resource utilization
  - CPU
  - Memory
  - Disk
- Shared-nothing VS Shared-disk.
- Multi-level Storage



Shared-nothing vs Shared-Disk



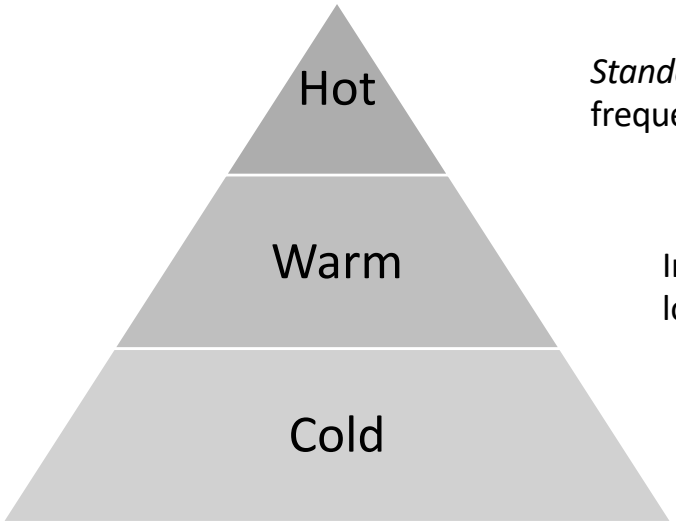
Shared-Nothing
Very large applications
High throughput writes
Need right partitioning strategy
Write-limited(2-phase commit)



Shared-Disk
Disaggregated Storage and Compute Architecture
Medium and small applications
Write-limited (coordinate lock)
Higher resource utilization

Resource Scheduling

- 3-level storage strategy



Standard Storage: on-line systems, require high availability and high frequency access, ensure 7*24 availability. e.g. Account data.

Infrequent access Storage: ensure medium-level availability. E.g. log data

Archive: planned visit, e.g. database backups.

Resource Scheduling

- 3-level storage strategy

Storage	Number of copies	Hardware
Standard storage	3 copies	SSD
Infrequent access storage	2 copies	SATA
Archive	1.3 copies	SATA

Hardware Customization:
1, customized high-density storage cabinets and low-speed disks
2, programmable power-off disks with low-power board

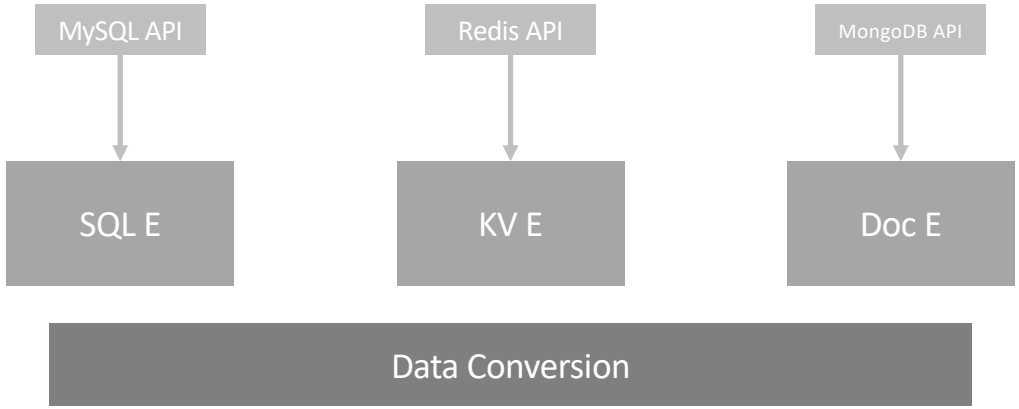
Automatic conversion:
Transfer data between different storage levels under different lifecycles

Multi-model

Why do we need multi-model?

- Hot Key Problems
  - Corporate Account or Hot SKU: very high concurrency
  - Relational database : Row lock contention, very low TPS
  - Key-value: customization, queuing

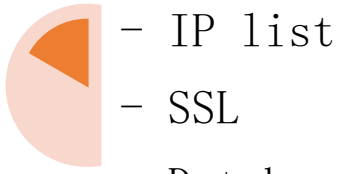
Multiple Data Model



SQL: standard
KV: highest performance
Doc: schemaless;

AI+Database

Data Security



Preventive control

- IP list
- SSL
- Database Firewall

Process control

- Transparent Data Encryption
- VPC

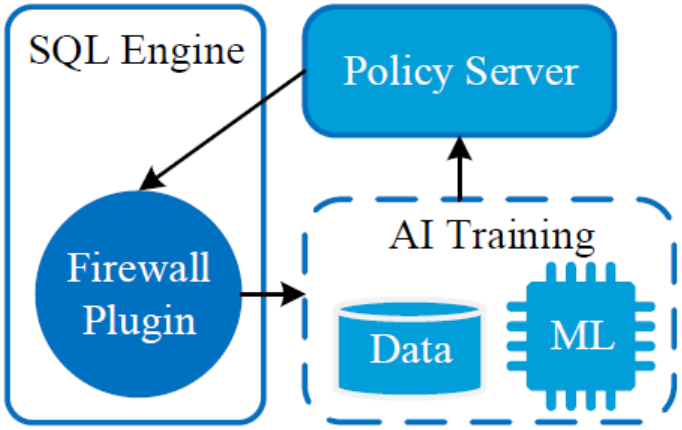
Post-action control

- DB Audit
- Log Audit

Why do we need AI + Security?

- More than 1,000,000 personal developers and enterprise customers.
- Too many kinds of applications, such as financial technology, smart retail, smart cities, smart transport, mobile payment, etc.
- Rule-based security policy

AI + Security



Risk: SQL injection, data security, etc.

Firewall: restrict access in accordance to the specified type of query, sql model, user, client IP
, etc.

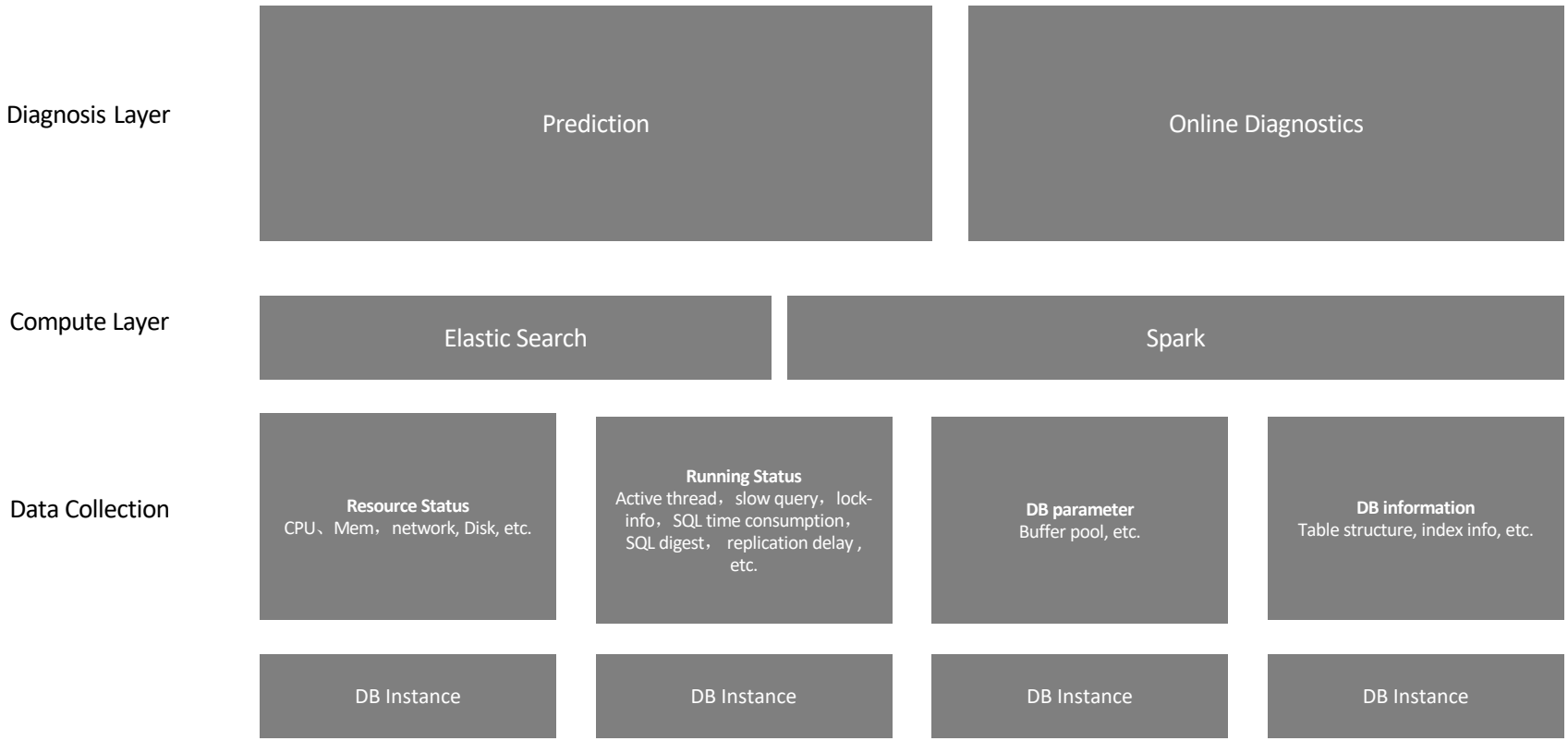
AI Training:

collect all query logs online;
tag all security events in the whole history
model training by AI teams

Why do we need AI + DBA?

- The growth of business VS the growth of DBA team
- fast troubleshooting
- Rapidly changing business VS Policy-based Monitoring
- Complex business scenarios VS volume performance estimating
- Bad SQL VS performance assurance

AI-DBA


Diagnosis Layer
Prediction
Online Diagnostics
Compute Layer
Elastic Search
Spark
Data Collection
Resource Status
CPU、Mem，network, Disk, etc.
Running Status
Active thread，slow query，lock-info，SQL time consumption，SQL digest， replication delay , etc.
DB parameter
Buffer pool, etc.
DB information
Table structure, index info, etc.
DB Instance
DB Instance
DB Instance
DB Instance

Outline

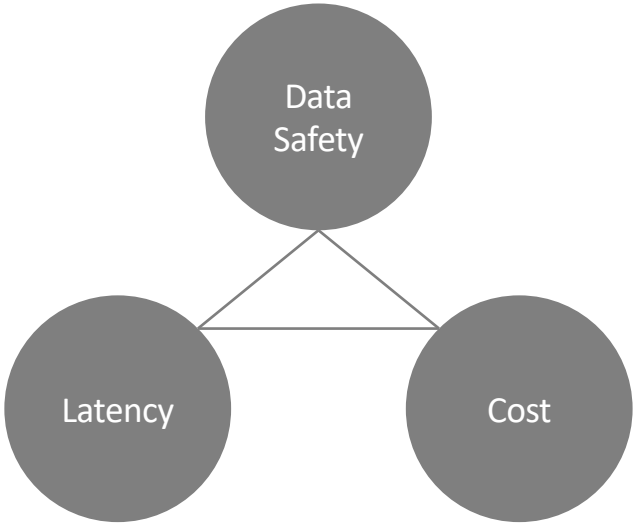
- Introduction
- System overview of TDSQL
- Conceptual Modeling on TDSQL
- Applications
- Conclusion

Deployment

Payment: Data Safety, Latency, Cost

WeChat Moment Feeds: Latency, Cost, Data Safety

Game: Latency, Cost, Data Safety

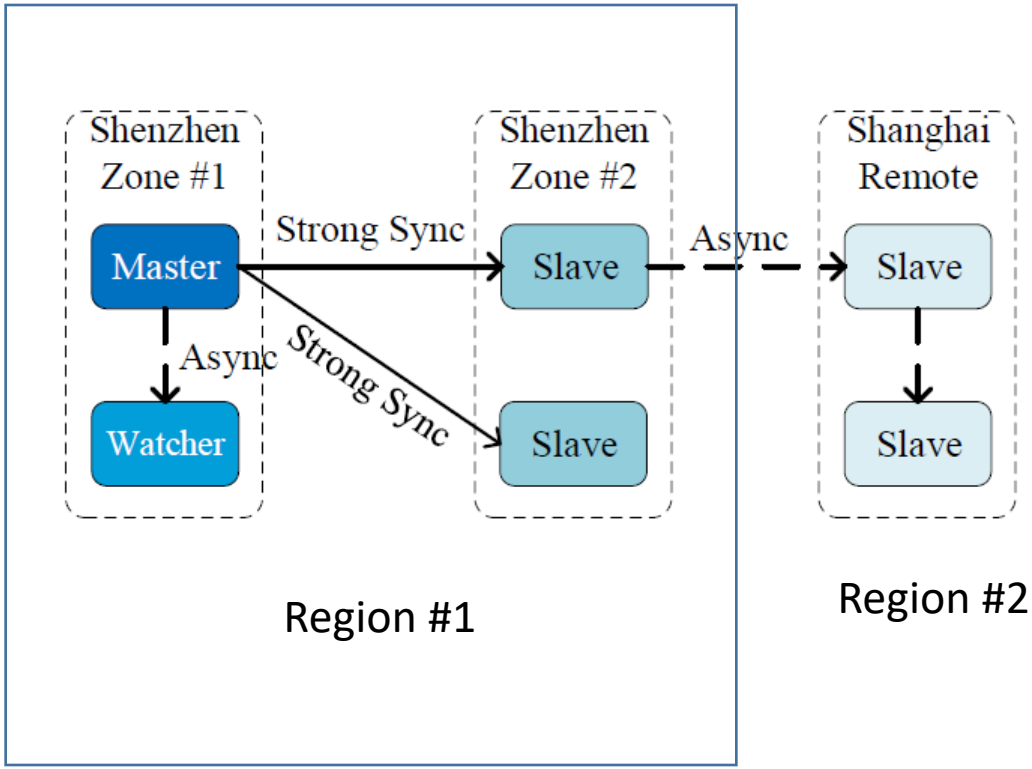


Deployment

1, When master data center is out of service, the cluster will be automatically switched to the standby data center.
Recovery Point Objective = 0
Recovery Time Objective less than 40s.

2, When only one server is out of service
   a) master
   b) slave
   c) watcher

3, When slave data center is out of service



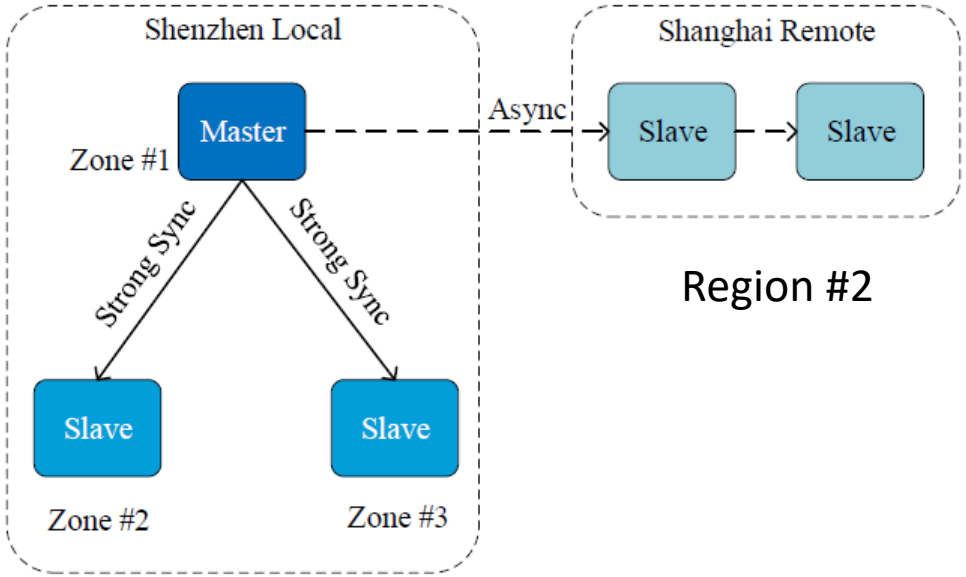
Deployment

1, Triple DZ in the same city simplify the synchronization policy and provide high data availability and consistency.

2, The failure of one data center does not affect data services.

3, The production cluster with triple-DZ provides multi-active services.

4, The entire city can be manually switched.

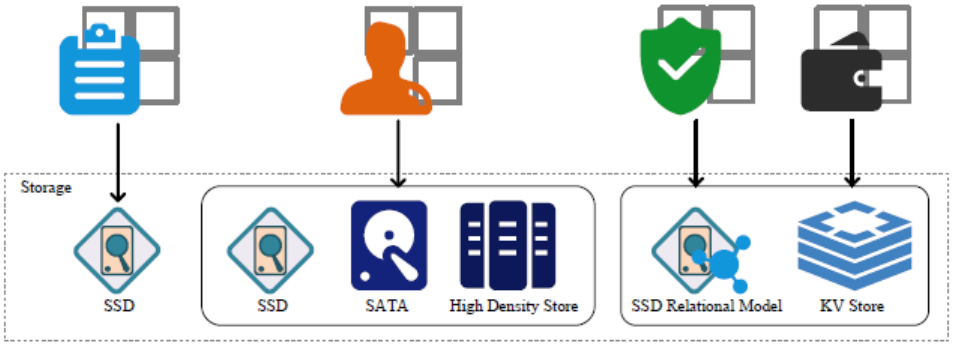


Resource scheduling

Order module: trading orders, data volume can be controlled, need high performance

Settlement Data: all historical data, very large volume, low cost, rarely accessed

Account Data: small volume, very high performance, randomly accessed

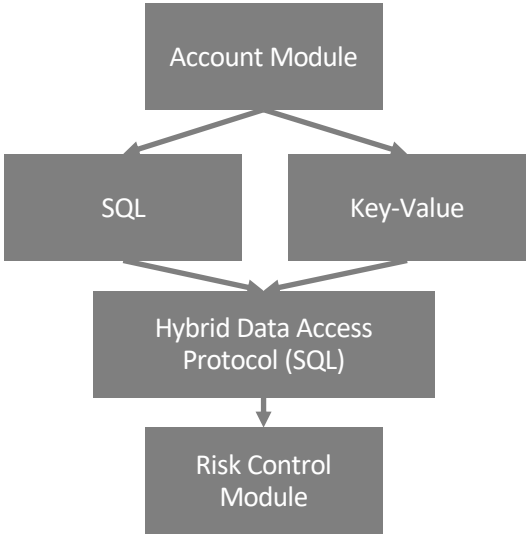


Hybrid data analysis

Personal Account Data: Relational model

Corporate Account Data: Key-value model

Risk Control Module: analyze both PAD and CAD



Outline

- Introduction
- System overview of TDSQL
- Conceptual Modeling on TDSQL
- Applications
- Conclusion

Conclusion

- One Size fits None
  - HTAP, OLTP, OLAP
  - Multi-model
- AI + Database

Thanks