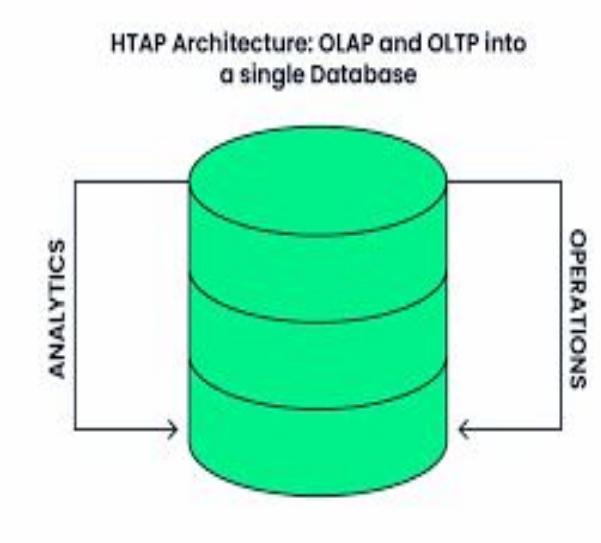
TiDB: A Raft-based HTAP Database

PingCAP

- Devaki Kulkarni

What is HTAP

- Hybrid Transactional/Analytical processing database
- OLTP + OLAP = HTAP
- Handles both OLAP and OLTP workloads
- **Goal:** to support real-time analytics on the most up-to-date transactional data
- Should provide resource isolation and guarantee consistency



Popular HTAP Databases

Requirements of an HTAP

- Low latency for transactional processing
- Real-time analytics
- Scalability
- Data Compression and indexing
- Schema flexibility

Approaches to build an HTAP system

- In-memory databases
- Separate storage engines
- Data virtualization
- Raft-based storage

What is Raft?

- Consensus algorithm for distributed systems
- Used to ensure group of nodes agree on same sequence of database operations and commits
- Consists of “leaders” and “followers”
- Elections occur to pick a leader who coordinates the distribution of log entries and final commit

Known challenges with Raft-based HTAP

- Achieving highly concurrent read/write operations
- Synchronization of logs into learner nodes with low latency
- Query Optimization of both transactional and and analytical queries

Role of Raft in TiDB

- Used to achieve consensus among nodes and replicate transaction logs across nodes
- Each partition of a table is called a Raft group, which consists of multiple nodes
- Raft group leaders are responsible for log replication
- Learner node:
  - A node that converts row format data to columnar format
- Raft group leaders coordinate distribution of logs to learner nodes

Architecture of TiDB

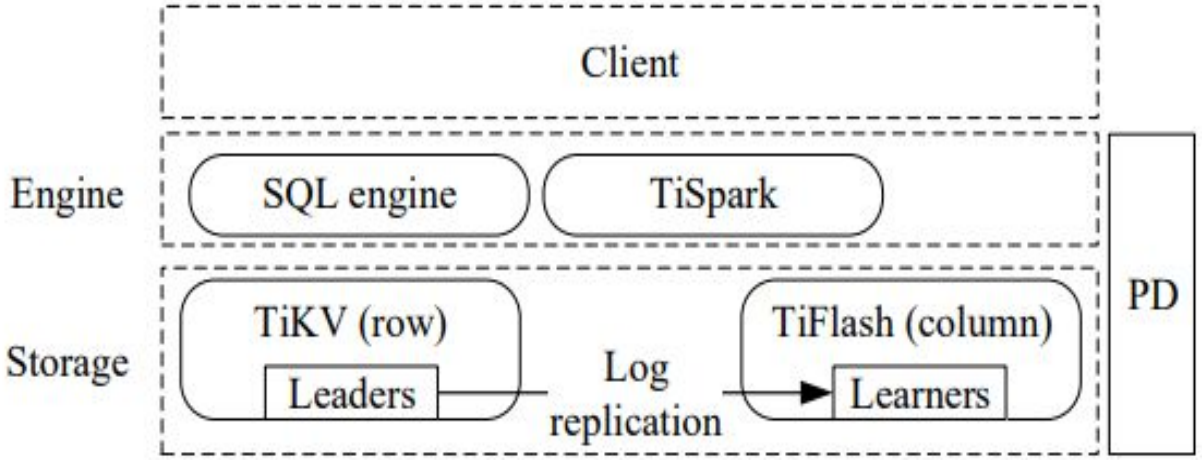


Figure 2: TiDB architecture

Components of TiDB architecture

1. TiKV
    - Is a distributed key-value store
    - Key-values are partitioned into Regions
    - Metadata for each TiKV server is stored in RocksDB, a persistent key-value store
2. TiFlash
    - Consists of Learner nodes that replicate data from Raft groups
    - Logs fetched from Raft groups are transformed from row-format tuples into columnar data
    - Learner nodes periodically fetch the current schema to avoid schema mismatch
    - DeltaTree used for reading and writing columnar storage with high throughput

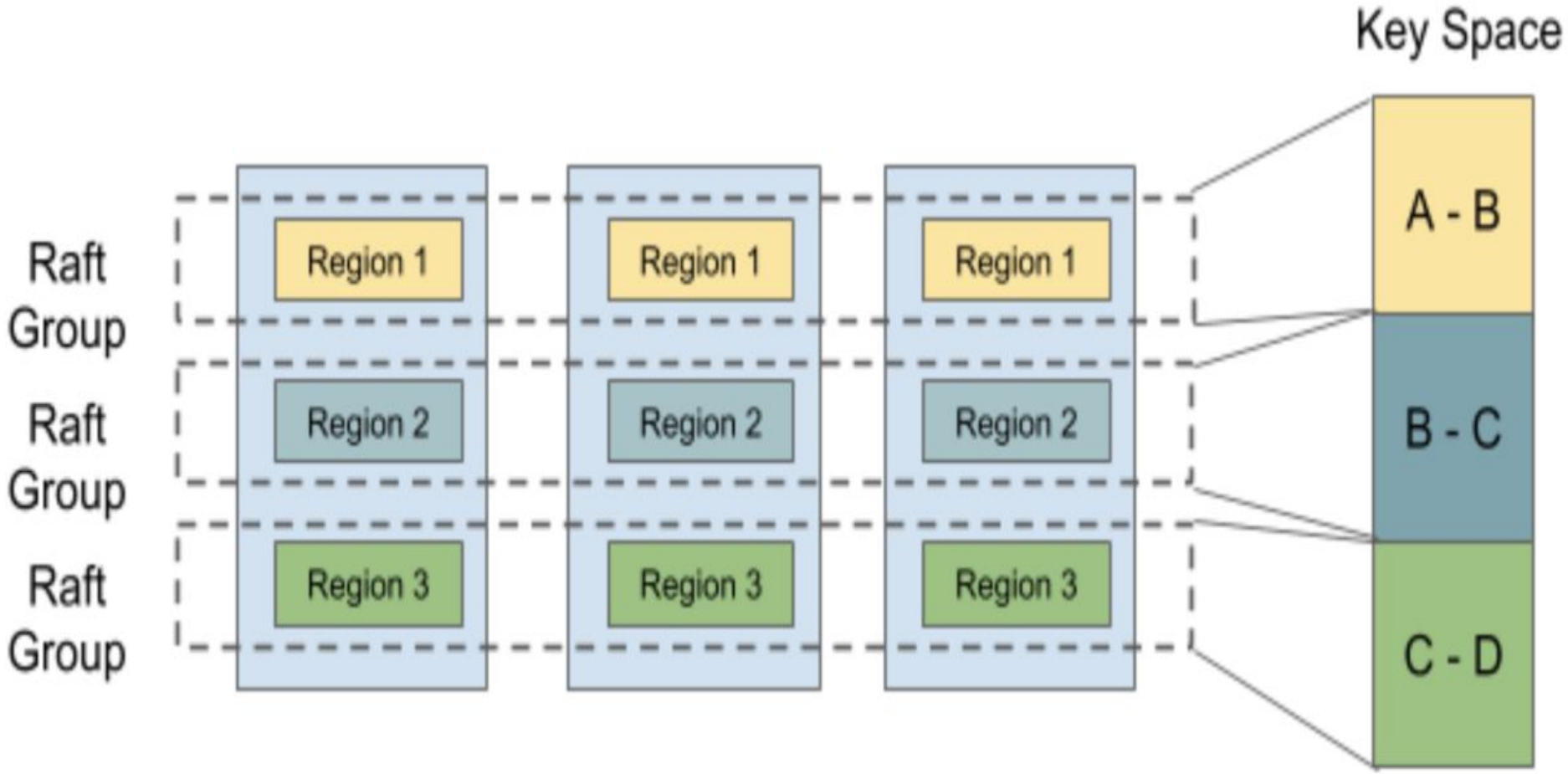
Components of TiDB architecture

3. Placement Device

- Used to keep track of multiple Regions within a Raft group
- Can move data across Regions
- Provides timestamps
- Balances workload by sending merge & split commands to TiKV

4. SQL Engine Layer

- Stateless and scalable
- Applies rule-based query optimizer to generate a logical plan
- Uses cost-based optimizer to generate a physical plan


Key Space
Raft Group
Raft Group
Raft Group
Region 1
Region 1
Region 1
Region 2
Region 2
Region 2
Region 3
Region 3
Region 3
A - B
B - C
C - D

How is a transactional R/W request handled by TiKV?

1. Each region leader receives a request from SQL Engine Layer
2. Leader appends request to the log
3. Leader sends new log entries to its followers, who also append entries to their logs
4. Leader waits for followers to respond. If quorum agrees, leader commits request and applied it locally
5. Result is send to client by leader and sequentially processes further requests

Optimizations for TiKV

1. Leader-follower optimizations
    a. Simultaneous log addition and distribution of log entries to followers
2. Accelerating read-requests from clients
    a. Read index approach
    b. Lease read approach
    c. Follower read
3. Balancing distribution of Regions over different servers by merging and splitting Regions

Transactional processing

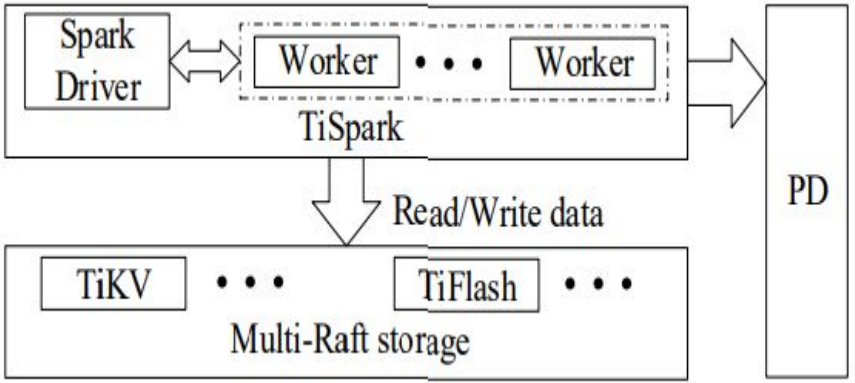
- Provides ACID guarantee along with RR and Snapshot Isolation
- Locks are stored in TiKV providing high scalability and availability.
- SQL engine and PD servers are scalable to handle OLTP requests
- TiKV uses 2PC commit along with optimistic and pessimistic locking

Analytical processing

- Query optimization:
  - Rule-based & Cost-based optimization
  - Skyline pruning algorithm
  - Physical plan is executed by SQL engine using pull iterator model
  - Coprocessor executes B trees of execution plan in parallel
  - Coprocessor can evaluate logical, logical operations, arithmetic operations, aggregations and TopN functions

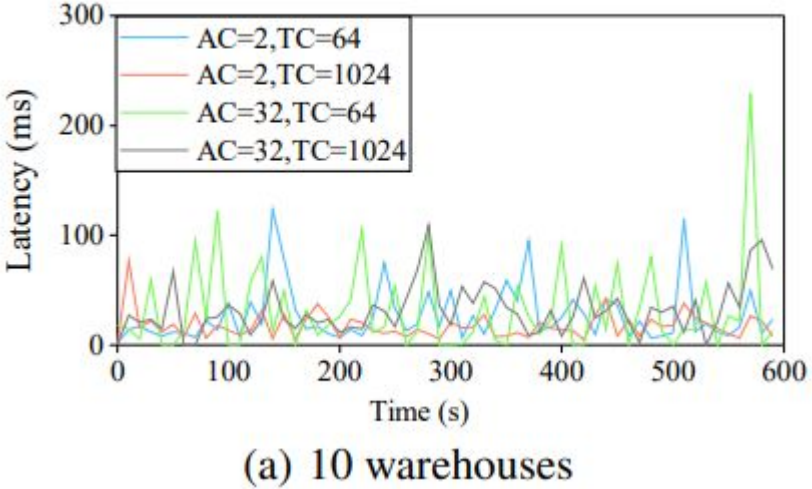
TiSpark

- Supports Hadoop ecosystem
- Supports ML libraries
- Provides concurrent reads from multiple TiKV regions
- Spark Driver keeps track of schema and metadata
- Worker nodes process data from TiKV and TiFlash

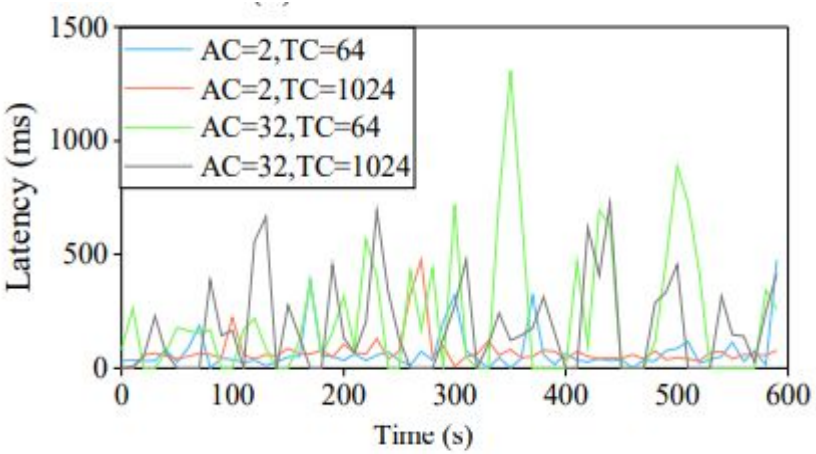


Performance evaluation

- Hybrid workload evaluated with CH-benCHmark



(a) 10 warehouses



(b) 100 warehouses

Questions?

Thank you!