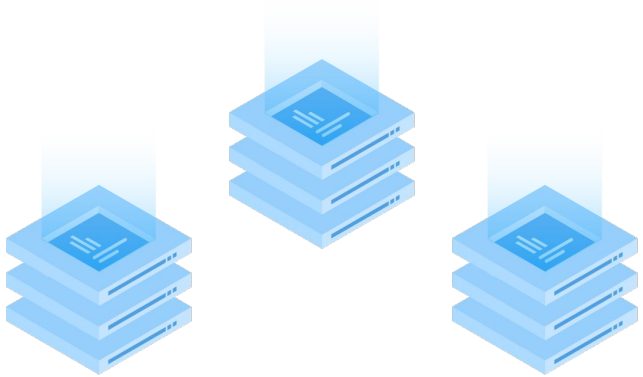
# Making HTAP Real

## TiFlash: A Native Columnar Extension for TiDB

PingCAP TiDB

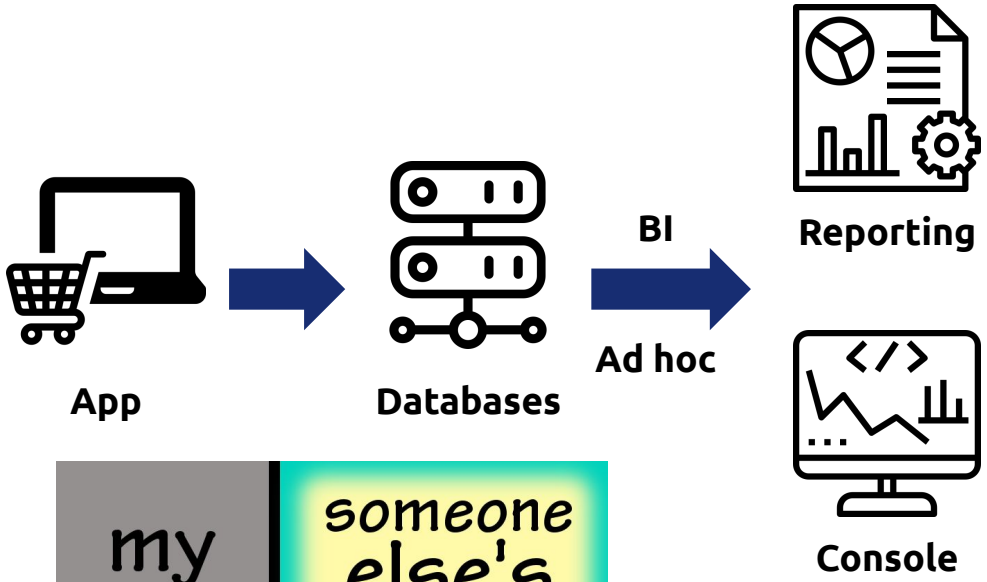


# About Me

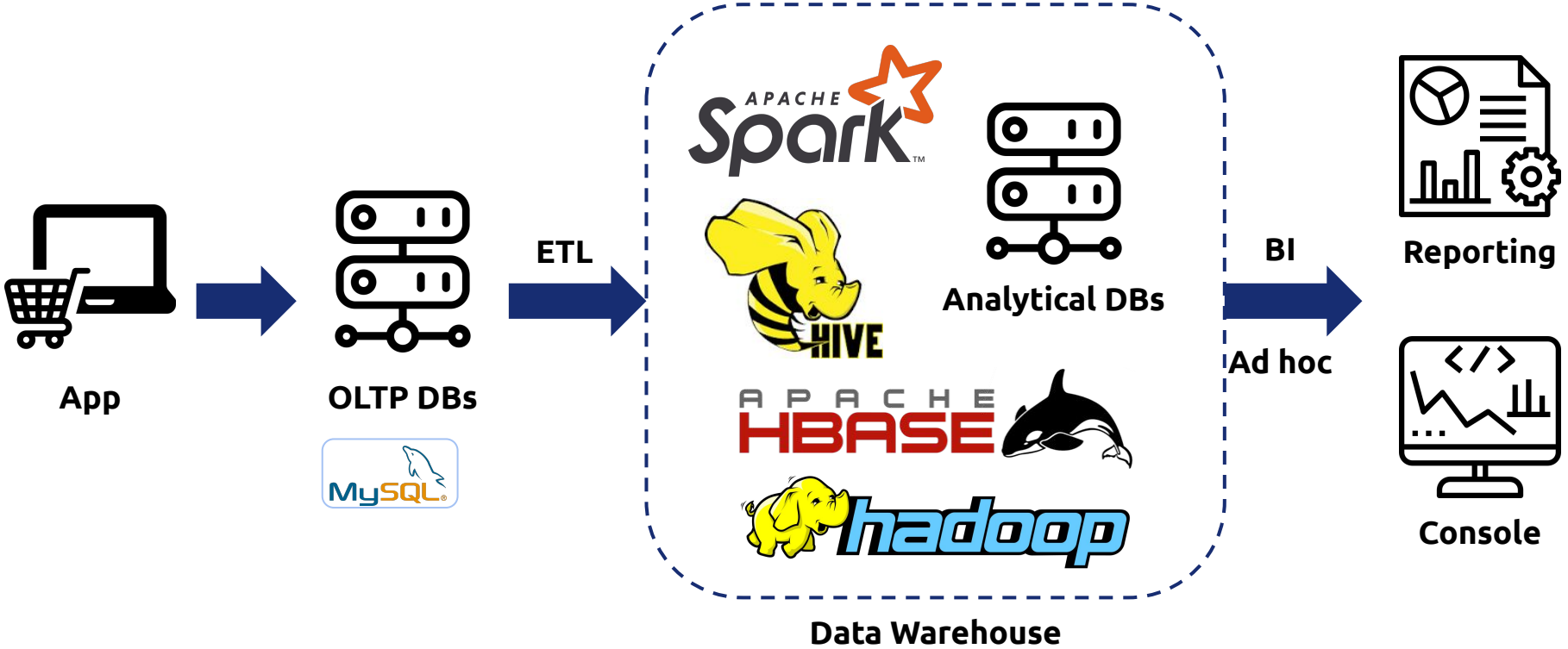
Dongxu (Ed) Huang, Co-founder & CTO @ PingCAP, a Distributed system engineer / Open source enthusiast / Guitar hobbyist

- Projects: TiDB / TiKV
- Location: Beijing / San Fransisco Bay Area
- Research interest: Distributed consenssus algorithm / transaction / storage engine design / data structure
- Email: h@pingcap.com
- Twitter: @dxhuang

# Data Platform - What Application Dev Think It Is



# Data Platform - What It Really Is



# Why is it so complicated?

- Different access patterns
  - OLTP
    - Short / point access to small number of records
    - Row-based format
  - OLAP
    - Large / batch process of subset of columns
    - Column-based format
- Workload interference/Resource isolation
  - OLAP queries can easily occupy large amount of system resources
  - OLTP latency / concurrency will be dramatically impacted

# A Popular Solution

- Different types of databases for different workload
  - OLTP specialized database for transactional data
  - Hadoop / analytical database for historical data

- Offload transactional data via ETL process into Hadoop / analytical database
  - Periodically, usually per day

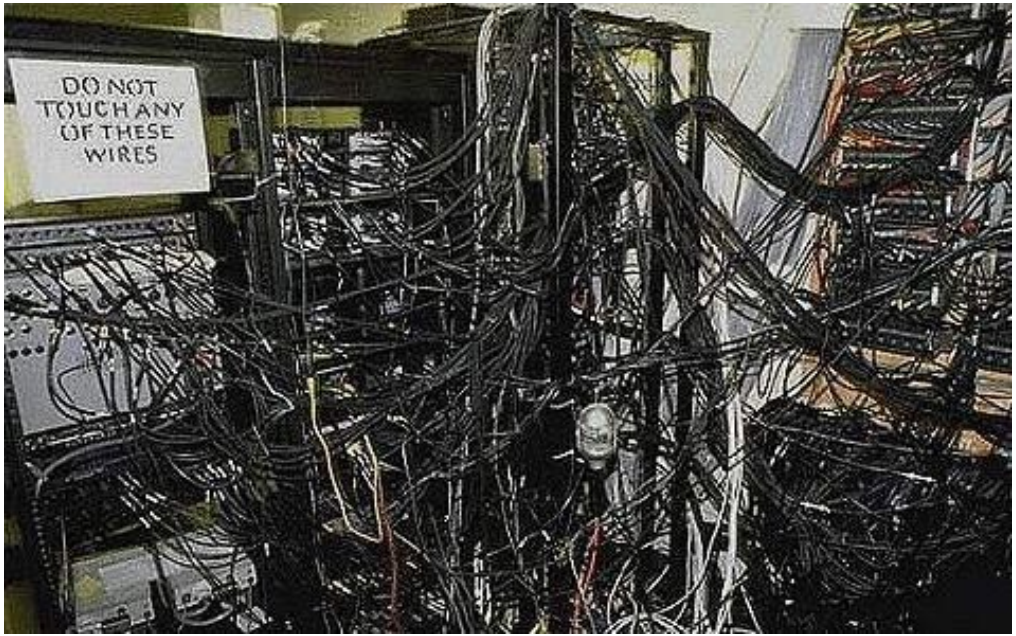
# Good enough!

# Really?

# Data processing stack today

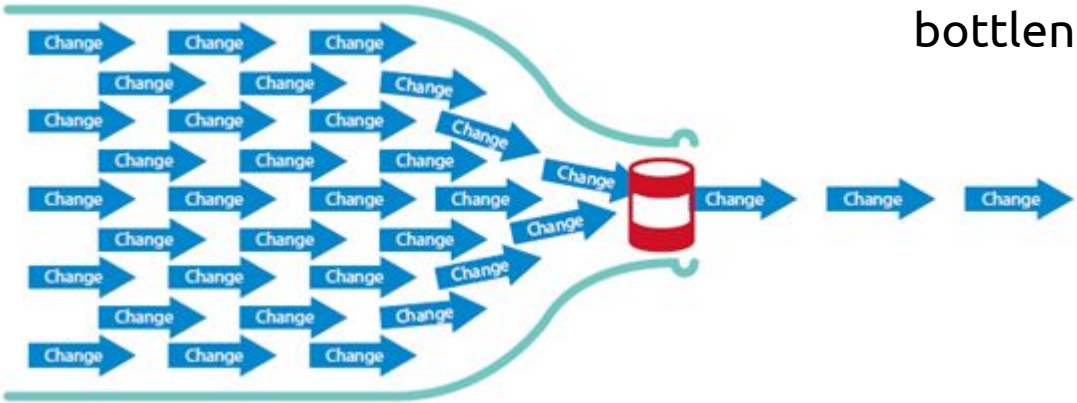

- Data cannot be seamlessly connected between different data stores

# Data processing stack today




- Adding a new data source is hard
- It is painful to maintain multiple systems.

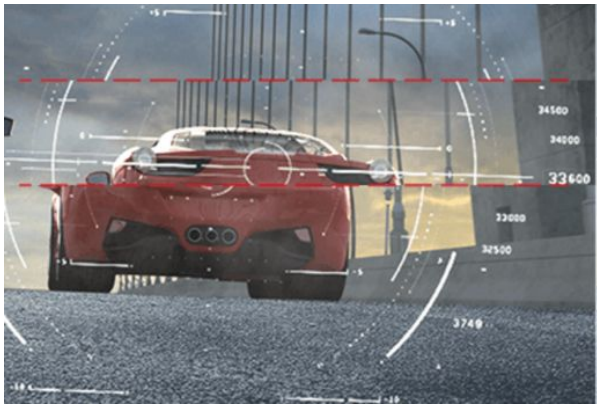
# Data processing stack today




- Data pipeline may easily become the bottleneck

# Data processing stack today

- The process of ETL usually loses the the transaction info

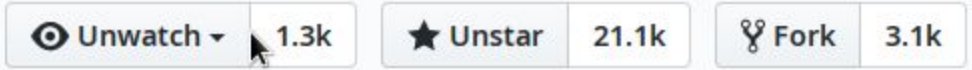



# A little bit about TiDB

- Full-featured SQL
  - MySQL compatibility
- ACID compliance
- HA with strong consistency
  - Multi-Raft
- Elastic scalability
- Open-source!

TiDB

A Distributed SQL Database




WASM TiDB: [https://play.tidb.io](https://play.tidb.io)

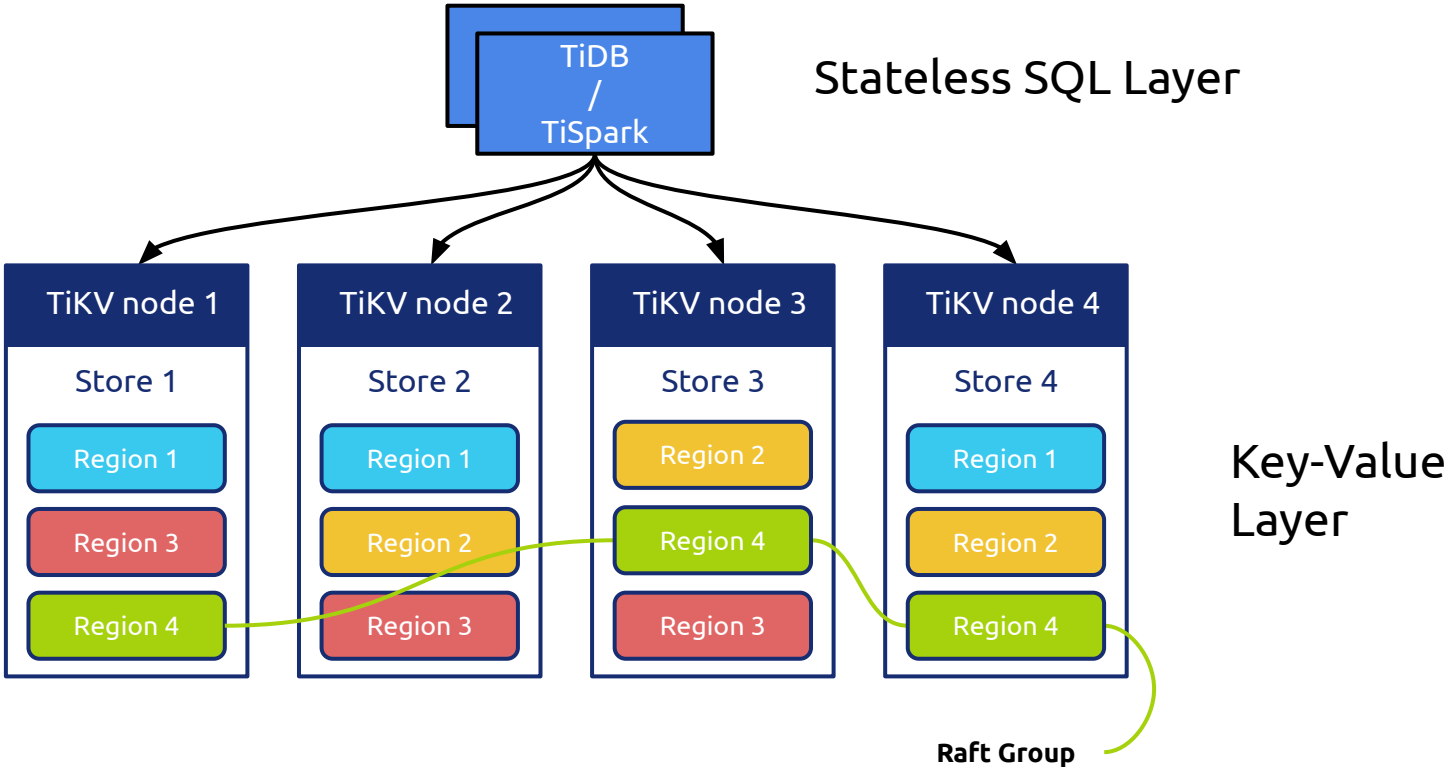
```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 113
Server version: 5.7.25-TiDB-v4.0.0-alpha-662-ge3b5577a0 MySQL Community Server (Apache License 2.0)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

# TiDB Architecture




“Several TB”

# Well, TiDB is a Row-based Database!

# If your database stores 100TB of data, and your database claims it supports full-featured SQL...

# Row-based or Column-based?

# Columnstore VS Rowstore

**Rowstore**

| id | name | age |
|---|---|---|
| 0962 | Jane | 30 |
| 7658 | John | 45 |
| 3589 | Jim | 20 |
| 5523 | Susan | 52 |

SELECT AVG(age) FROM emp;

**Columnstore**

| id |
|---|
| 0962 |
| 7658 |
| 3589 |
| 5523 |

| name |
|---|
| Jane |
| John |
| Jim |
| Susan |

| age |
|---|
| 30 |
| 45 |
| 20 |
| 52 |

# Columnstore VS Rowstore

- Columnstore
  - Bad for small random reads and random writes
  - Suitable for analytical workload
  - Efficient CPU utilization using vectorized processing
  - High compression rate
- Rowstore
  - Good for random reads and random writes
  - Researched and optimized for OLTP scenario for decades
  - Cumbersome for analytical workload

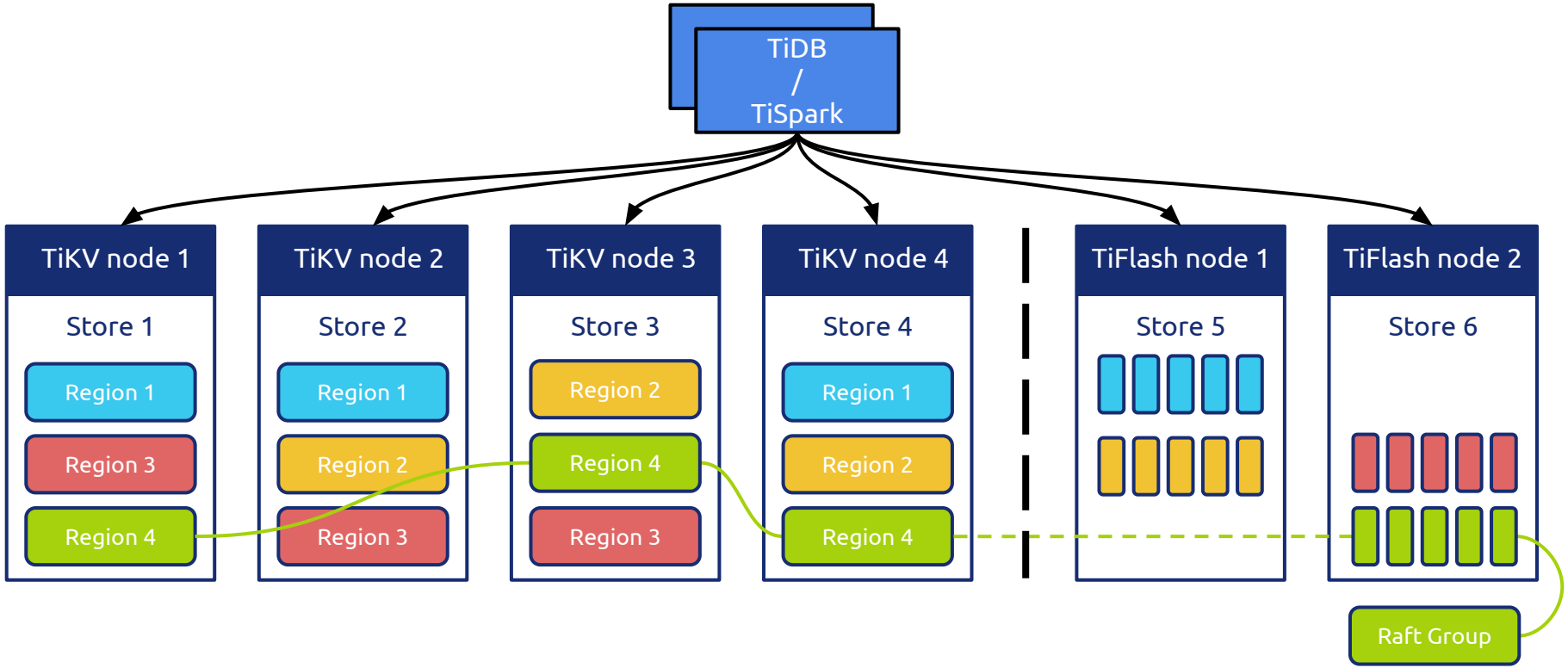
# Why not both?

# TiFlash

# What Is TiFlash?

- An extended analytical engine for **TiDB**
  - Columnar storage and vectorized processing
  - Partially based on ClickHouse with tons of modifications
  - Enterprise offering (May open source in the future)
- Data synchronization via extended **Raft** consensus algorithm
  - Strong consistency
  - Trivial overhead
  - Raft learner, a non-voting member in the Raft group
- Strict workload isolation to eliminate the impact on OLTP
- Native integration with TiDB

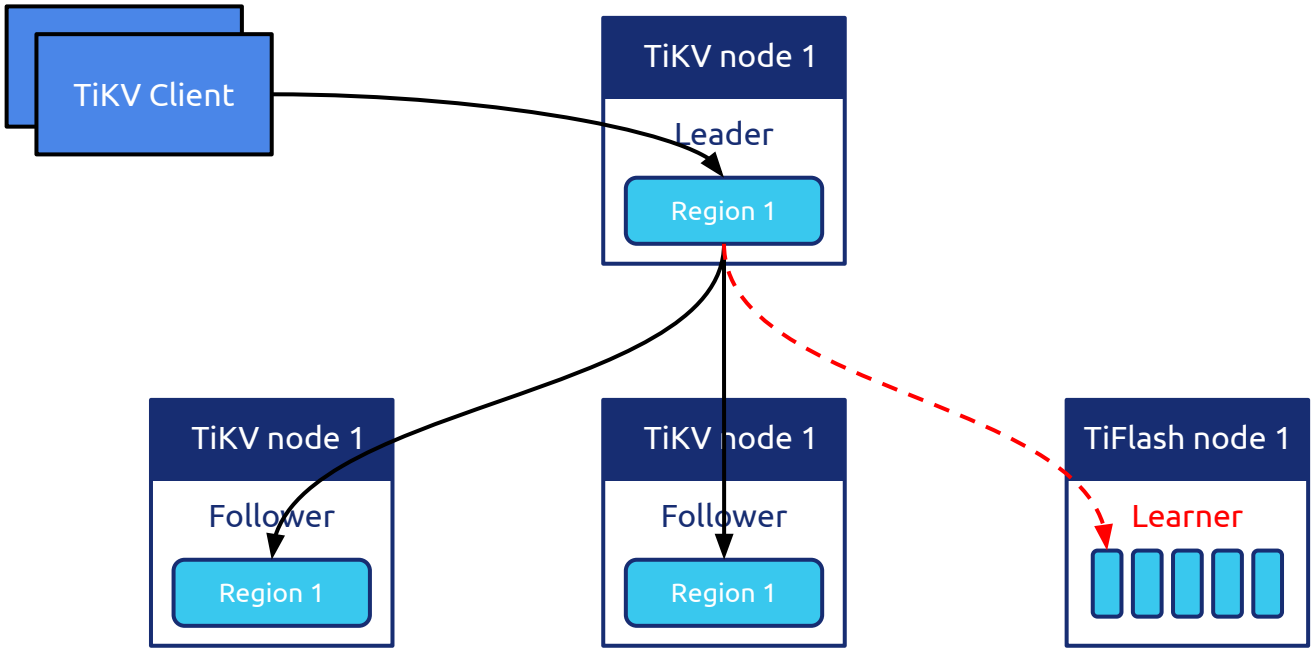
# TiDB with TiFlash Architecture



# Low-cost Data Replication

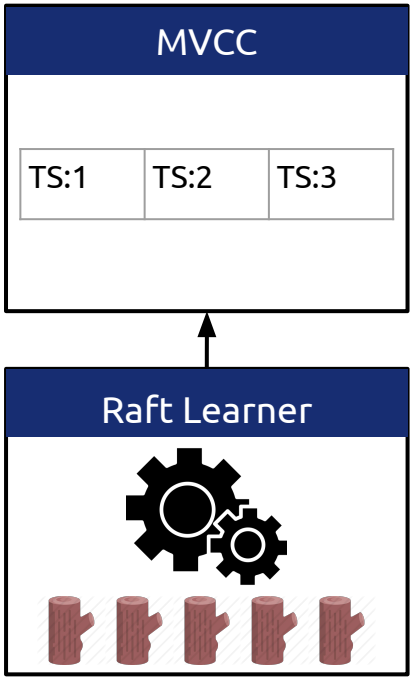
- Data is replicated to TiFlash via **Raft Learner**
  - Non-voting member
  - Extended Raft consensus algorithm
  - Async replication
  - Almost zero overhead to OLTP workload

# Low-cost Data Replication

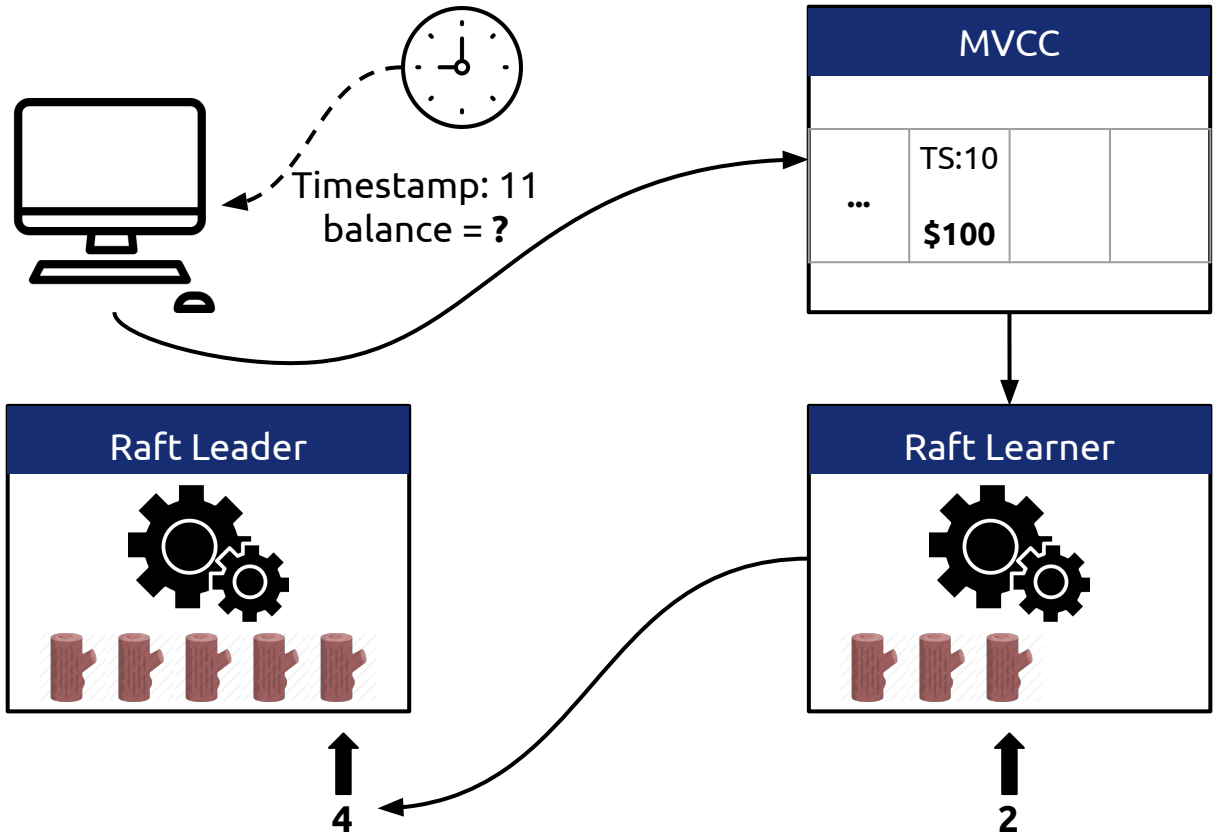


# Strong Consistency

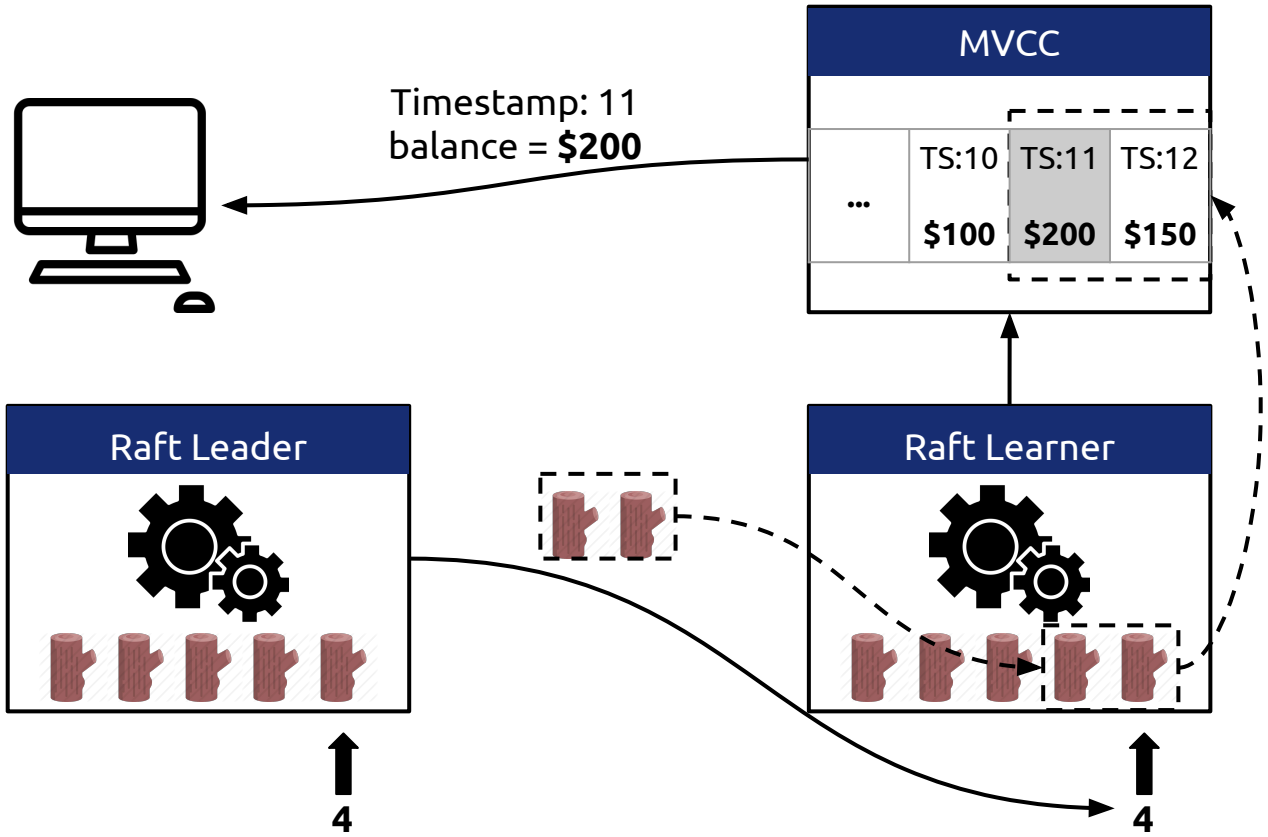
- Logically the same view as in rowstore
  - Same data
  - Same isolation level (SI)
- TiFlash keeps casual consistency via async replication
  - 99.99…% in-sync
  - 0.00…1% out-of-sync
- Read operation guarantees strong consistency
  - Learner Read
  - Multiversion concurrency control (MVCC)



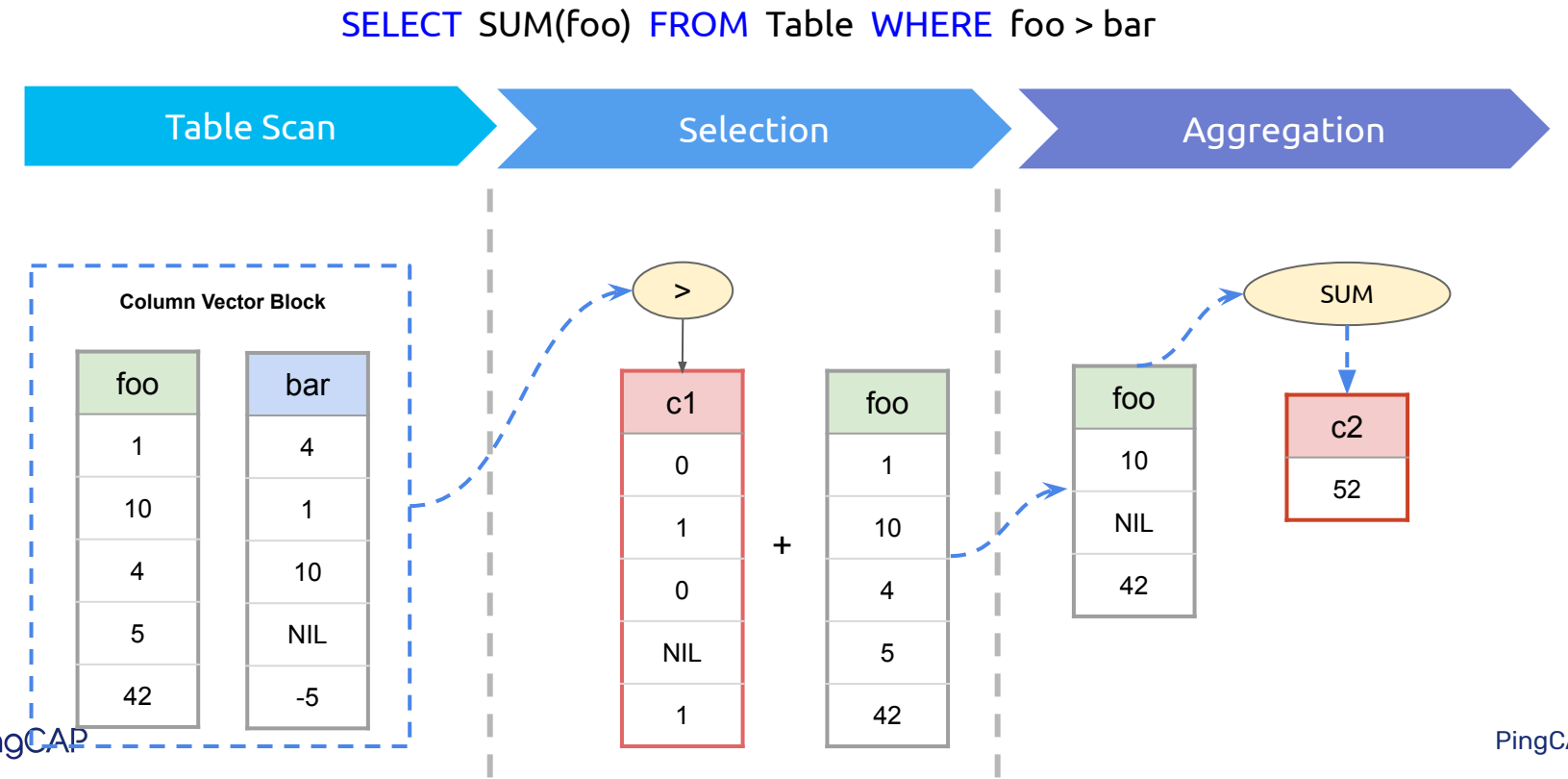
# Learner Read + MVCC



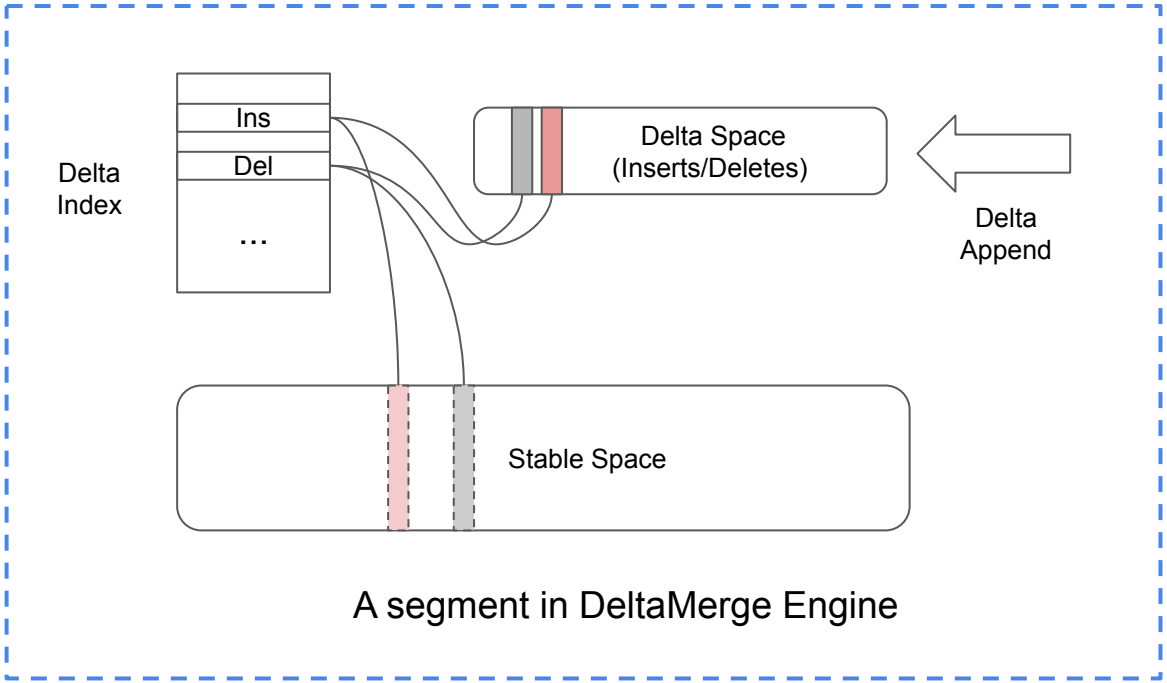
# Learner Read + MVCC



# Vectorized Processing

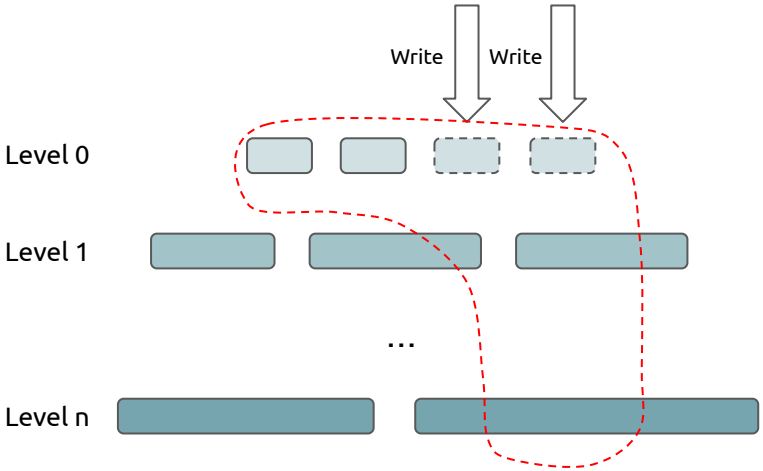


# DeltaMerge Engine



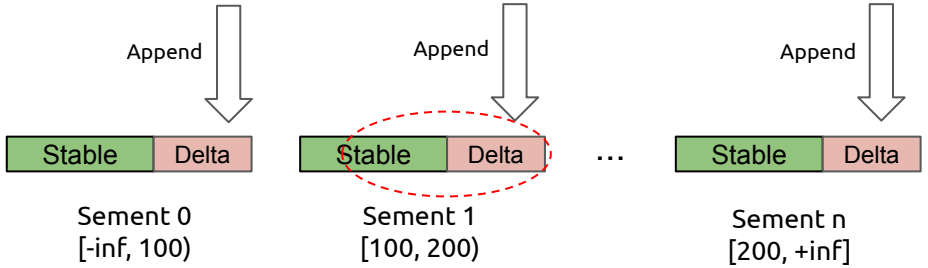
# DeltaMerge Engine

SELECT ... WHERE x BETWEEN (150, 160)




LSM-Tree

VS




DeltaMerge

# TiFlash is beyond columnar storage

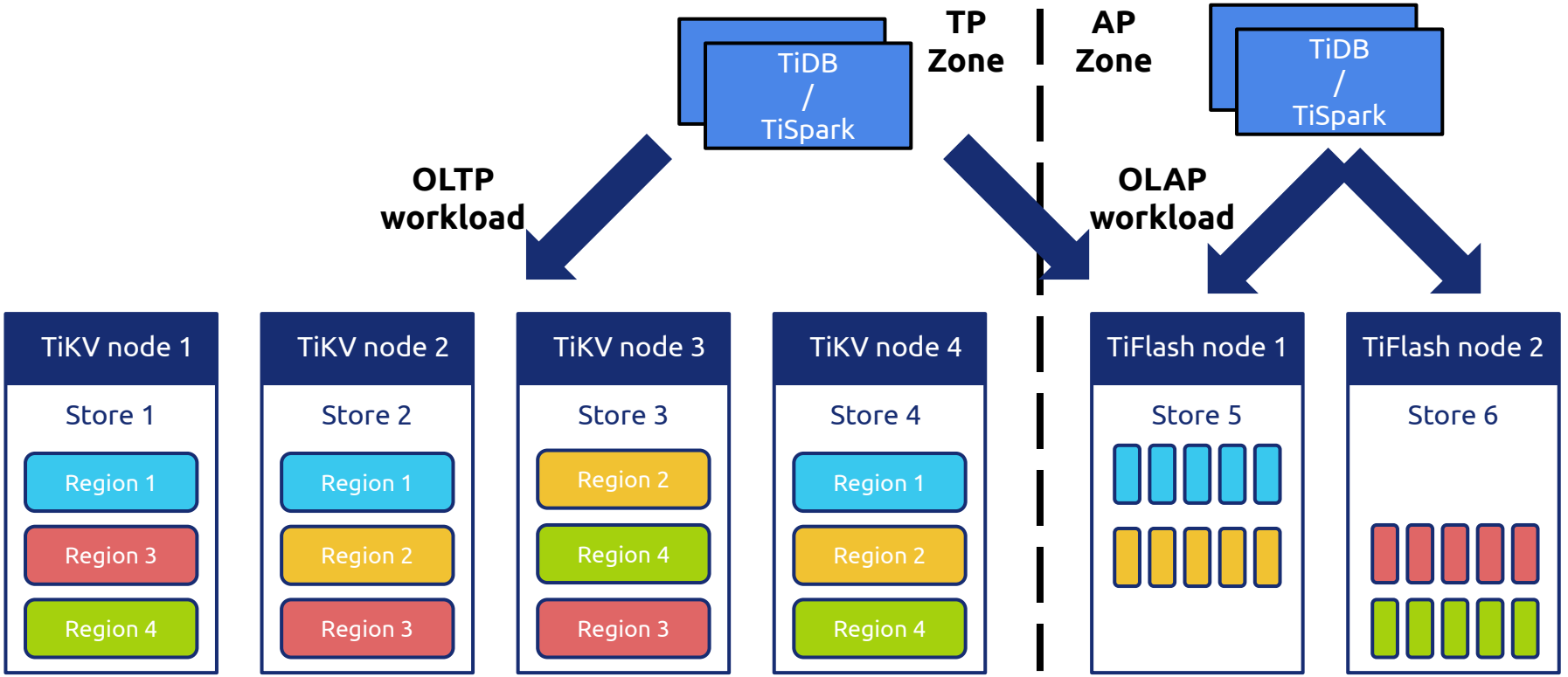
# Scalability

- TiDB relies on **Multi-Raft** for scalibility
  - One command to add / remove node
  - Scaling is fully automatic
  - Smooth and painless data rebalance
- TiFlash fully inherits these abilities

# Isolation

- Perfect resource isolation to prevent workload interference
- Dedicated nodes for TiFlash
- Nodes are clustered into “zone”s
  - TP Zone
    - TiKV nodes, for OLTP workload
  - AP Zone
    - TiFlash nodes, for OLAP workload

# Isolation

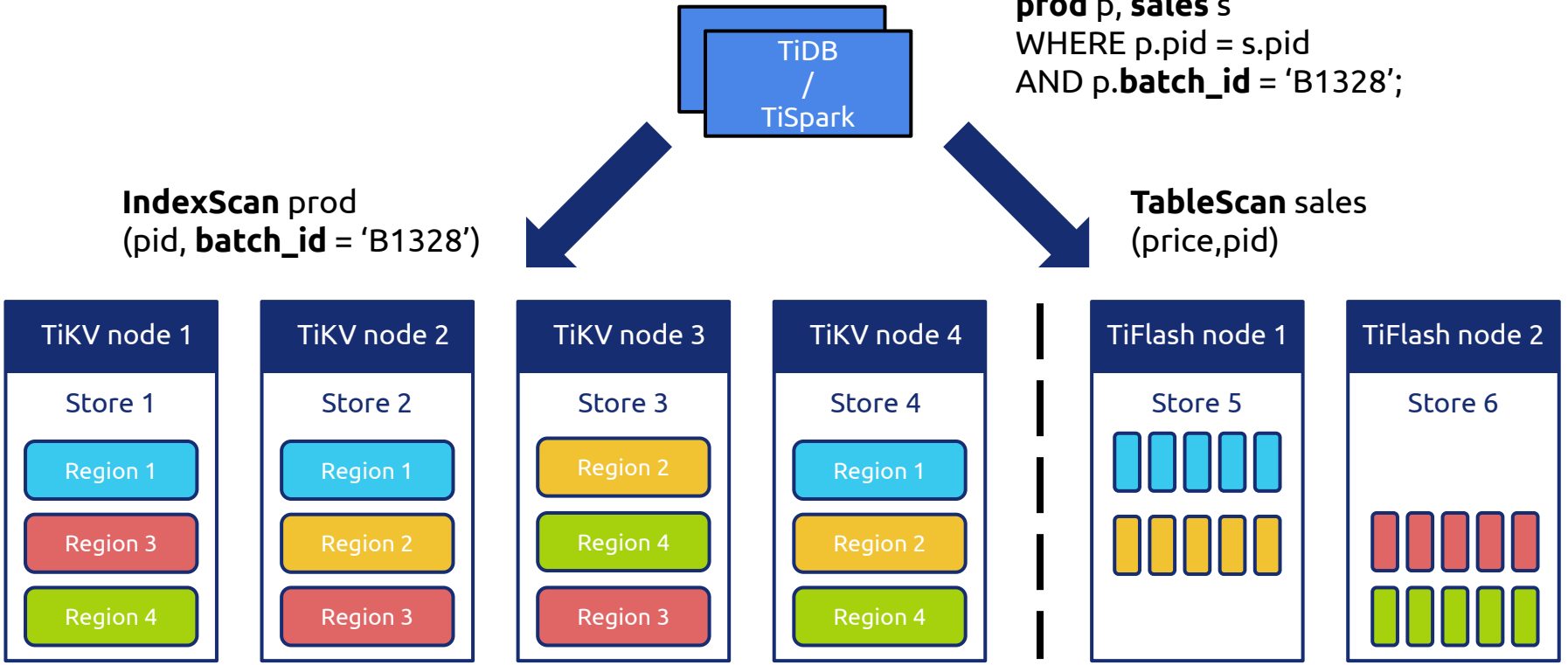

TiDB / TiSpark
TP Zone
AP Zone
TiDB / TiSpark
OLTP workload
OLAP workload
TiKV node 1
Store 1
Region 1
Region 3
Region 4
TiKV node 2
Store 2
Region 1
Region 2
Region 3
TiKV node 3
Store 3
Region 2
Region 4
Region 3
TiKV node 4
Store 4
Region 1
Region 2
Region 4
TiFlash node 1
Store 5
TiFlash node 2
Store 6

# Integration

- TiDB / TiSpark might choose to read from either side
  - Based on cost
  - Columnstore is treated as a special kind of index
- Upon TiFlash replica failure, read TiKV replica transparently
- Join data from both sides in a single query

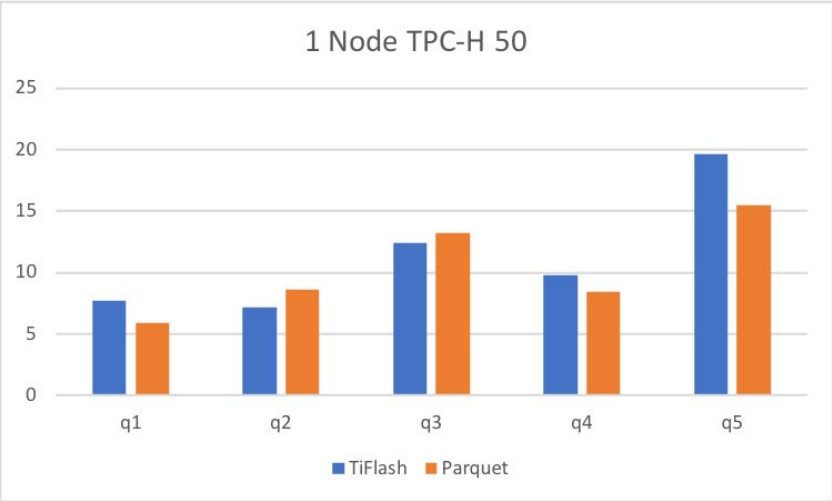
# Integration

SELECT AVG(s.price) FROM
**prod** p, **sales** s
WHERE p.pid = s.pid
AND p.**batch_id** = 'B1328';



# Performance

- Comparable performance against Parquet format
  - Underlying storage format supports Multi-Raft + MVCC
- Benchmark against Apache Spark 2.3 on Parquet
  - Pre-POC version of TiFlash + Spark



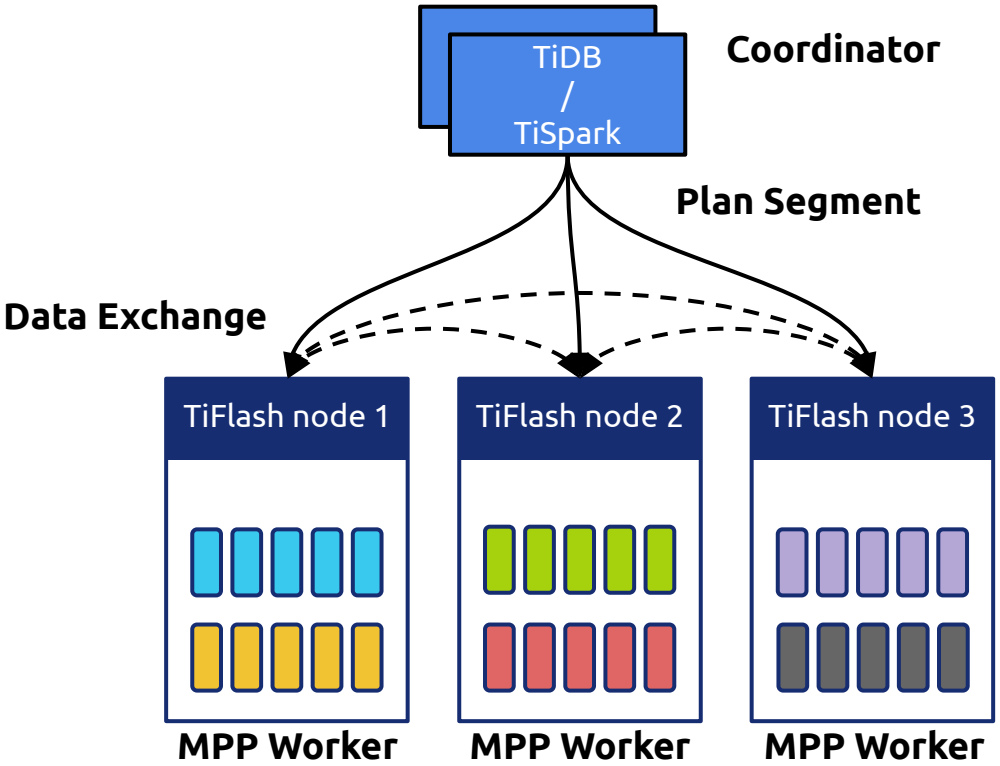
# WIP Items

- Native MPP
- Direct Write as Columnar format

# Massively Parallel Processing (MPP) Support

- TiFlash nodes form a MPP cluster by themselves
- Full computation support will
  - Further speed up TiDB by pushing down more computations
  - Speed up TiSpark by avoiding writing disk during shuffle

# MPP Support

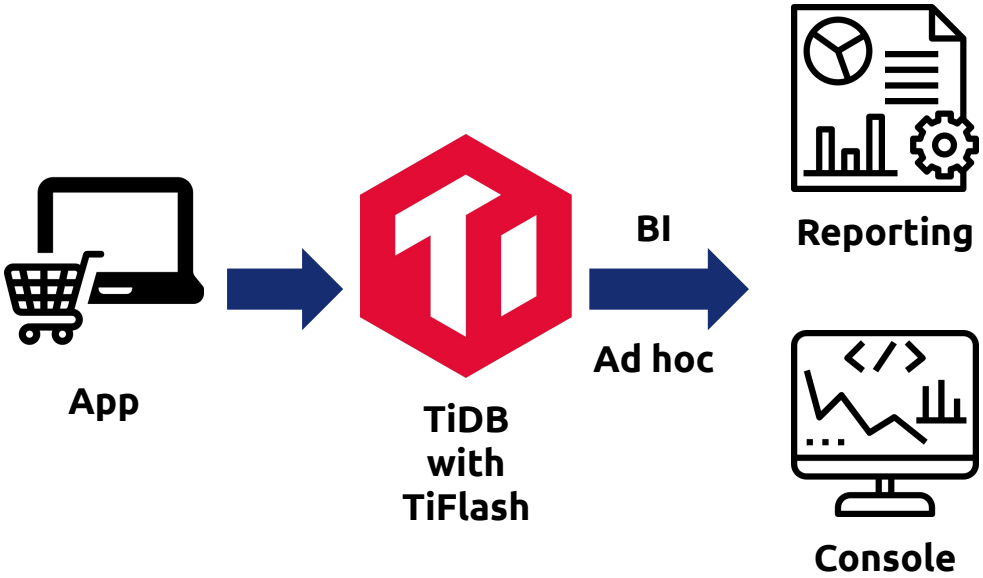

TiDB
/
TiSpark
Coordinator
Plan Segment
Data Exchange
TiFlash node 1
TiFlash node 2
TiFlash node 3
MPP Worker
MPP Worker
MPP Worker

# Direct Write as Columnar format

- For now, TiFlash is only mirroring data from TiKV
  - User needs to write data into TiKV (rowise format) first
- We are working on direct columnar write support
  - Faster batch write for data warehousing application
  - No mandatory 3 TiKV replicas anymore
  - Cheaper historical data storage

# TiDB Data Platform

# Data Platform - What It Could Be



*“What happened yesterday?”*

**VS**

*“What’s going on right now?”*

# Roadmap

- 1st round Beta now
  - With columnar engine and isolation ready
    - Access only via Spark
- Nov 2nd round Beta
  - With CBO and TiDB integration ← *We're here.*
- GA with TiDB 4.0 (around 2020 1Q)
  - Unified coprocessor layer
    - Ready for both TiDB / TiSpark
    - Cost based access path selection

PingCAP

# Thank you!

Email: huang@pingcap.com
info@pingcap.com