TiDB’s Distributed SQL Architecture: For Scale and Reliability

Introduction

Sunny Bains
Architect, PingCAP

- Working on database internals since 2001.
- Was MySQL/InnoDB team lead at Oracle.

TiDB’s unique value

- Easy to setup and start
- MySQL 8.0 compatible
- Scalable by Design
- Disaggregated Compute and Storage
- Multi-tenant ready
- Versatile
- Reliable
- Open source

中国农业发展银行

CICC 中金公司

NETSCAN

ele·style

杭州银行 BANK OF HANGZHOU

万达普惠

colopl

陆金所 Lufax.com

中国南方电网

中国银联 China UnionPay

国信证券 GUOSEN SECURITIES

ENN新奥

ZTO 中通快递 ZTO EXPRESS

招商银行 CHINA MERCHANTS BANK

CTW

AJU

Bank 中国光大银行

中国银河证券 CHINA GALAXY SECURITIES

拉卡拉

LONGFOR 龙湖

广发银行 CHINA GUANGFA BANK

YumChina

SAKURA internet

中国移动 China Mobile

AMBER

浦发银行 SPD BANK

agora

中国人寿 CHINA LIFE

dailymotion

Cygames

PLAID

CERTIK

LINE

menu

平安科技 PING AN TECHNOLOGY

中国平安人寿保险 PING AN LIFE INSURANCE OF CHINA

Shopee

Bolt

翼支付

DMM.com

Opera Ads

北京银行 BANK OF BEIJING

Chaintool

Square

CyberAgent.

U-NEXT

优速快递 UC EXPRESS

亿联银行 YILLION BANK

Micoworks

小红书

スマレジ

BIGO

CREDIT 维信金科

浙商银行

bookmyshow

中国经济信息社

贝壳金服

BAO ZUN

盖娅互娱

SF EXPRESS 顺丰速运

易宝支付 YEEPAY.COM

株式会社ネットスターズ

SHAREit

WonderPlanet

China unicom 中国联通

CAPCOM

中国银行 BANK OF CHINA

WeBank 微众银行

ninjavan

pony.ai

Agenda

01 Design Fundamentals

TiDB Architecture

02 Resource Control

Empowering Consolidated Workloads with Precision Resource Allocation. DXF.

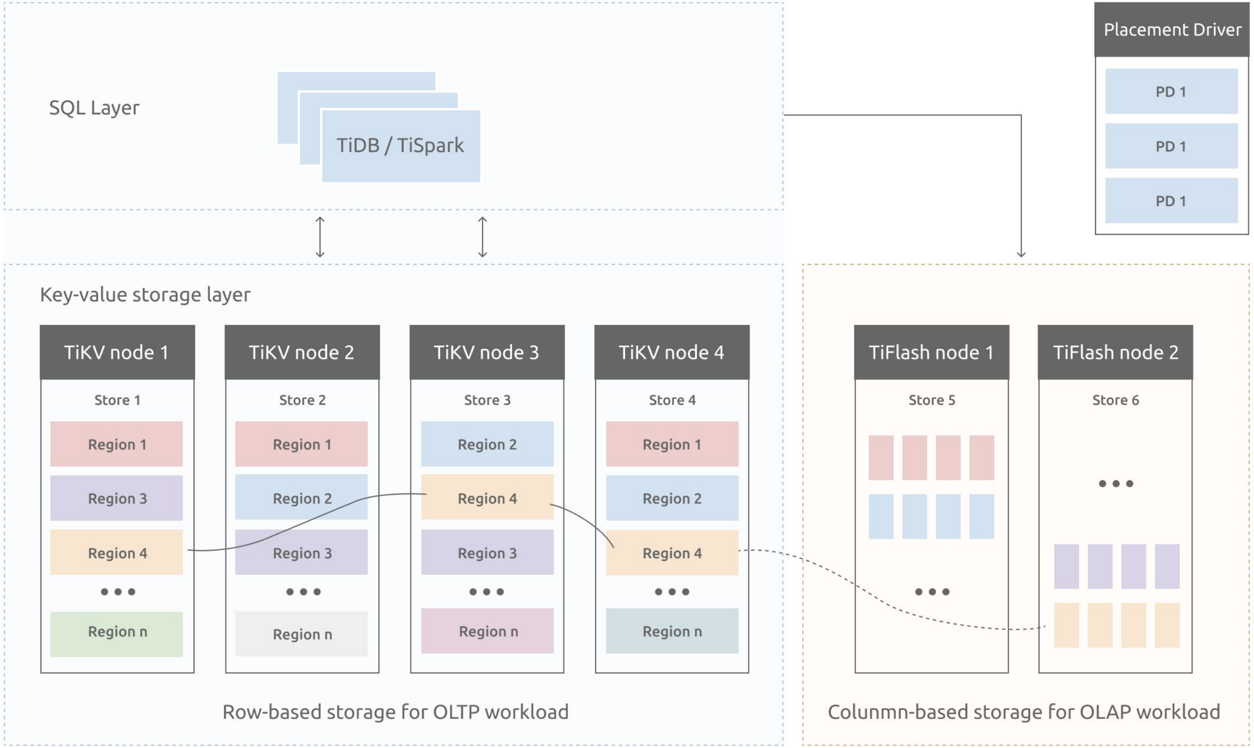
03 Online DDL

Enhancing Database Agility with Lightning-Fast Schema Changes

04 Tools

TiDB's wide range of tools for managing your databases

Reference Architecture



- **OLTP** and **optional** OLAP
- **Raft** for consensus
- Data consistency
- Configurable region size
- Fault tolerance, across AZ

TiDB Region

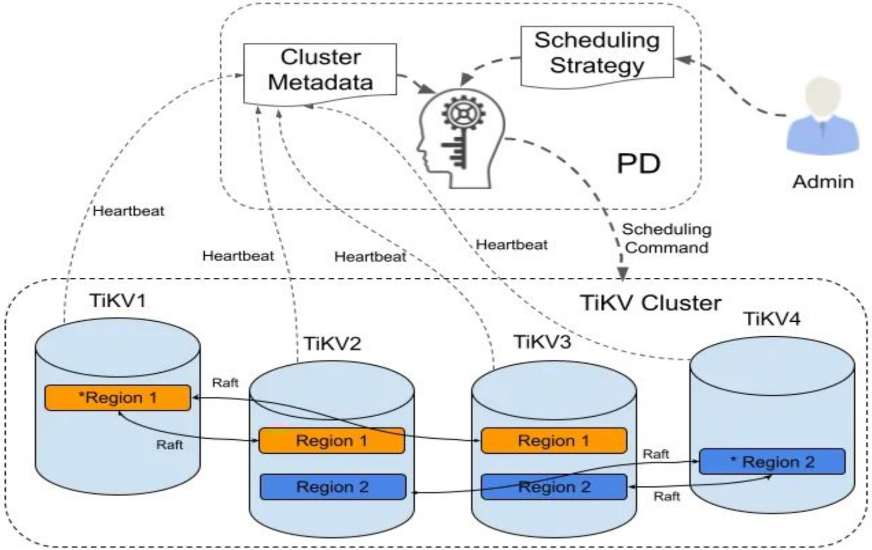
A **Region** is TiKV's **logical** scale unit

- Operations such as load balance, scale-out, scale-in are all based on **region**
- **Regions** are replicated using the Raft consensus protocol
  - A replicated **region** is called a Raft group
  - **Regions** are spread across the nodes in the cluster
  - A single node contains many **Regions**
  - **Regions** are stored on RocksDB, there is one instance of RocksDB per Node.
  - Rows in a **Region** are ordered

Placement Driver [PD]

PD is the meta-data server for the cluster and coordinates the entire cluster

PD is stateless, stores the global state in etcd
PD's stateless design allows it to achieve HA using etcd



PD Overview

Generates the start and commit Timestamp (TS) of distributed transactions

Handles region distribution and node failures

- Dynamic balancing and rebalancing, spread the love evenly
- Workload balancing, identify and avoid hotspots dynamically

Handles cluster configurations

- Facilitates migration of region replicas to added nodes
- Automatically manages online/offline state transitions of nodes

Multi-Zone deployment and disaster recovery

PD Cluster Monitoring

PD collects information at two levels of granularity

- Node level
  - Total and free disk capacity
  - The number of Regions
  - Data writing speed
  - The volume of sent/received Snapshots (used for data replication)
  - Node overload status, CPU monitoring
  - Label information (a set of hierarchical Tags)

- Region level heartbeat messages (Raft consensus protocol related messages)
  - The location of the Leader and Followers
  - The number of disconnected Followers
  - Data reading and writing qps

PD Scheduling strategies

PD policies are settable by the administrators

- Replication factor constraint
- Replica placement constraint
  - Policy to force spread of replicas over node/rack/dc/zone
    - e.g., ensure that the replicas are spread geographically
    - A disconnected node rejoins the cluster leading to excessive replicas
    - Ensure raft (region) leaders are spread evenly across the nodes
- Balanced space utilization across the cluster
  - Using the free and used storage on all the nodes
- Hotspot detection and mitigation
  - Using the CPU and read/write throughput metrics sent by the nodes in the cluster
- Governor for scheduling
  - Controls scheduling rate by monitoring ongoing operations, by default it tends to conservative. The speed can be adjusted via the administration interface.

PD Placement Policies

PD placement policies are settable using SQL

Create and set a placement policy

```
CREATE PLACEMENT POLICY myplacementpolicy
    PRIMARY_REGION = "us-east-1" REGIONS = "us-east-1, us-west-1";

CREATE TABLE t1 (a INT) PLACEMENT POLICY = myplacementpolicy;

CREATE TABLE t2 (a INT);

ALTER TABLE t2 PLACEMENT POLICY = myplacementpolicy;
```

Modify a placement policy

```
ALTER PLACEMENT POLICY myplacementpolicy FOLLOWERS = 4;
– Create 5 replicas [one leader and 4 followers]
```

Drop a placement policy

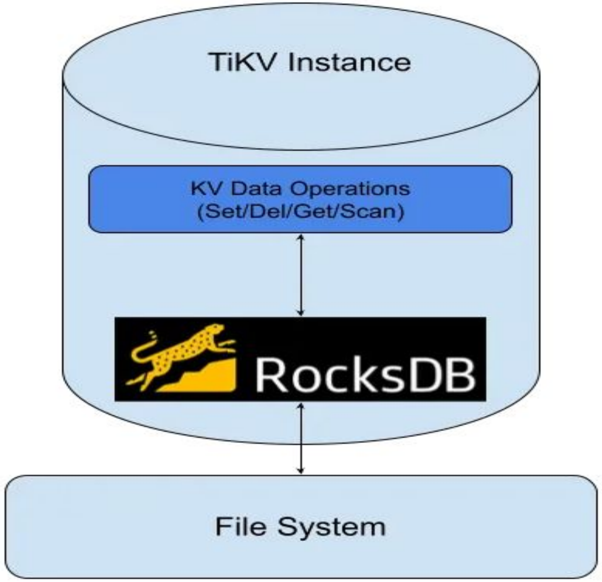
```
DROP PLACEMENT POLICY myplacementpolicy;
```

TiKV - Distributed storage engine

CNCF Graduated Project. Written in Rust.

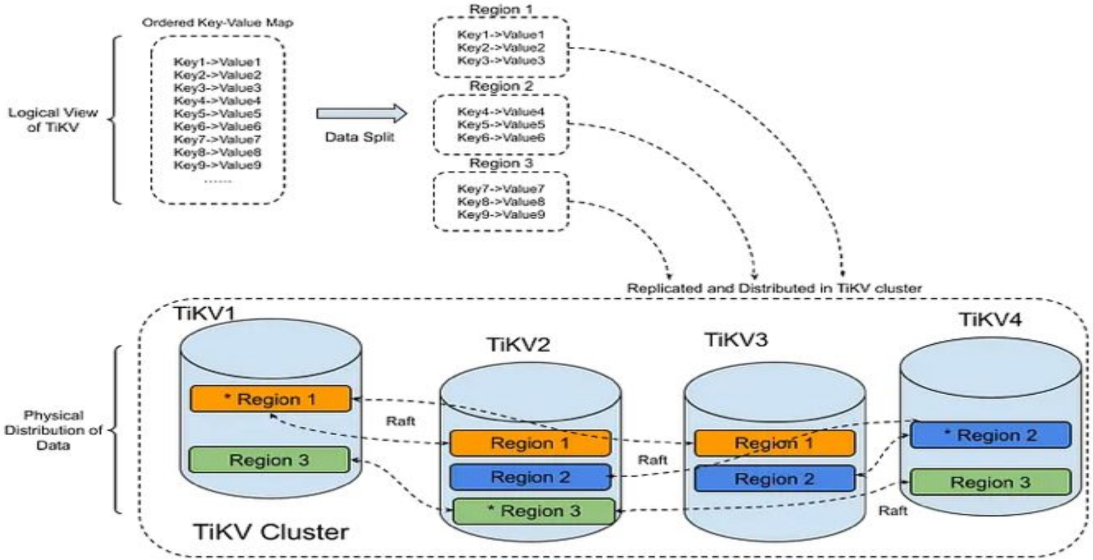
TiKV provides the following services
- Store and retrieve the data
- Replication and fault tolerance
- Data distribution across the storage cluster
- Distributed transaction processing

You can visualize it as a large distributed and ordered hash map that is designed for high performance and reliability.



TiKV - Data Storage Example

Example to illustrate how TiKV partitions and manages the data



TiKV - Coprocessor

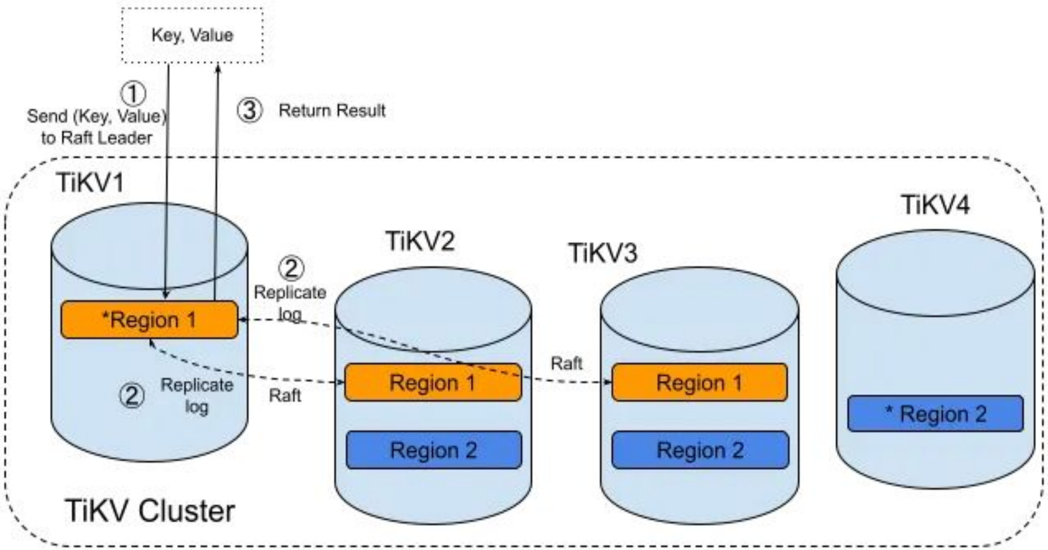
The TiKV Coprocessor supports the following executors
The names are self explanatory, they can be chained together

- **Table scanner**
- **Index scanner**
- **Selector (Table scanner | Index scanner)**
    - Performs a filter, mostly for where.
- **Aggregator (Table scanner | Index Scanner | Selector)**
    - Performs an aggregation (e.g. count(*), sum(E))
- **Top N elements (Table scanner | Index scanner | Selector)**
    - Sorts the data and returns the top N matches, for example, order by C limit 10

TiKV - Raft Consensus Protocol

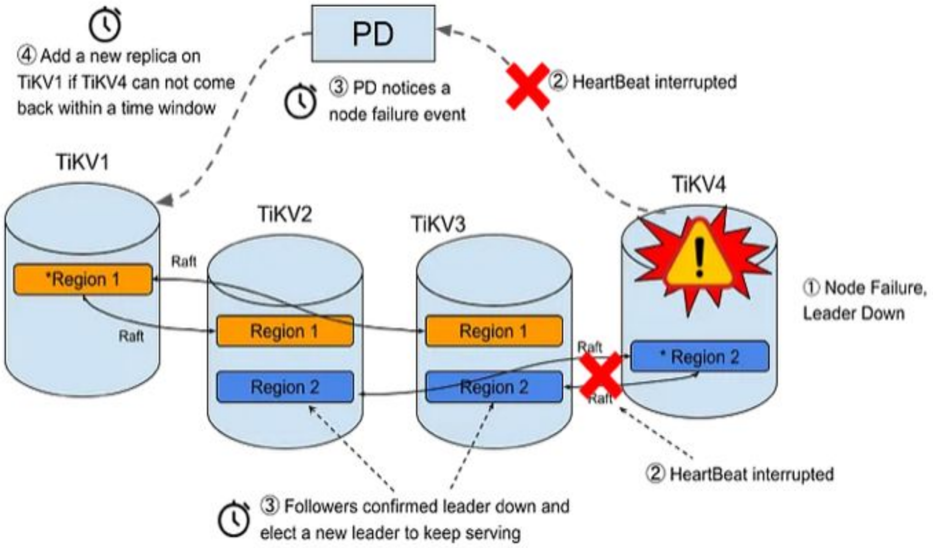
The core idea of Raft is to elect a leader and all writes then go through the leader.

The data is not considered durable until a majority of the nodes in the cluster acknowledge the write.



TiKV - Handling Failures

Hardware and network failures are a fact of life

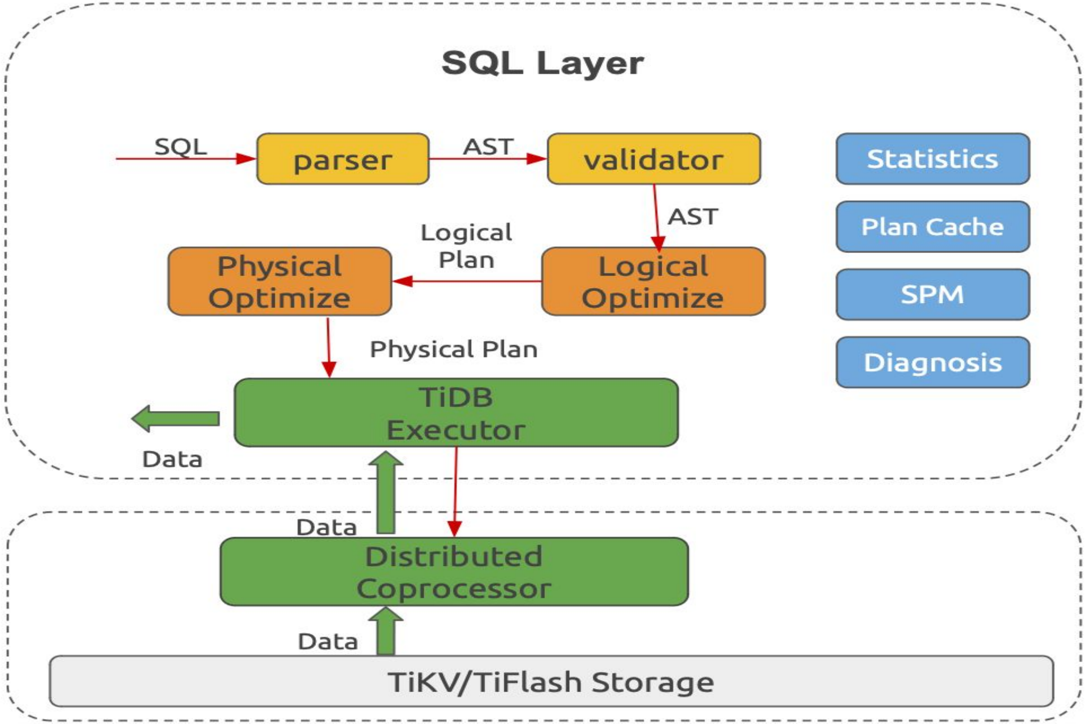


Distributed Transactions in TiDB

- TiDB supports Read-Committed and Snapshot Isolation levels
  - The Snapshot Isolation is mapped to MySQL/InnoDB's Repeatable Read
- TiDB uses an optimized version of the Percolator algorithm for distributed transactions
- A transaction requires a start time stamp and a commit timestamp
  - PD is responsible for handing out these timestamps
  - These timestamps are used in TiDB's MVCC implementation
- Async commit in TiDB
  - The SQL nodes are the Txn Coordinators (TC)
  - The TiKV nodes are the participants
  - Works well when the transaction write set is small and Phase II time dominates
- Supports 1PC Commit Optimization
  - If transaction only updates a non-index column of a record
  - Or, Inserts a record without a secondary index,
  - Only involves a single Region

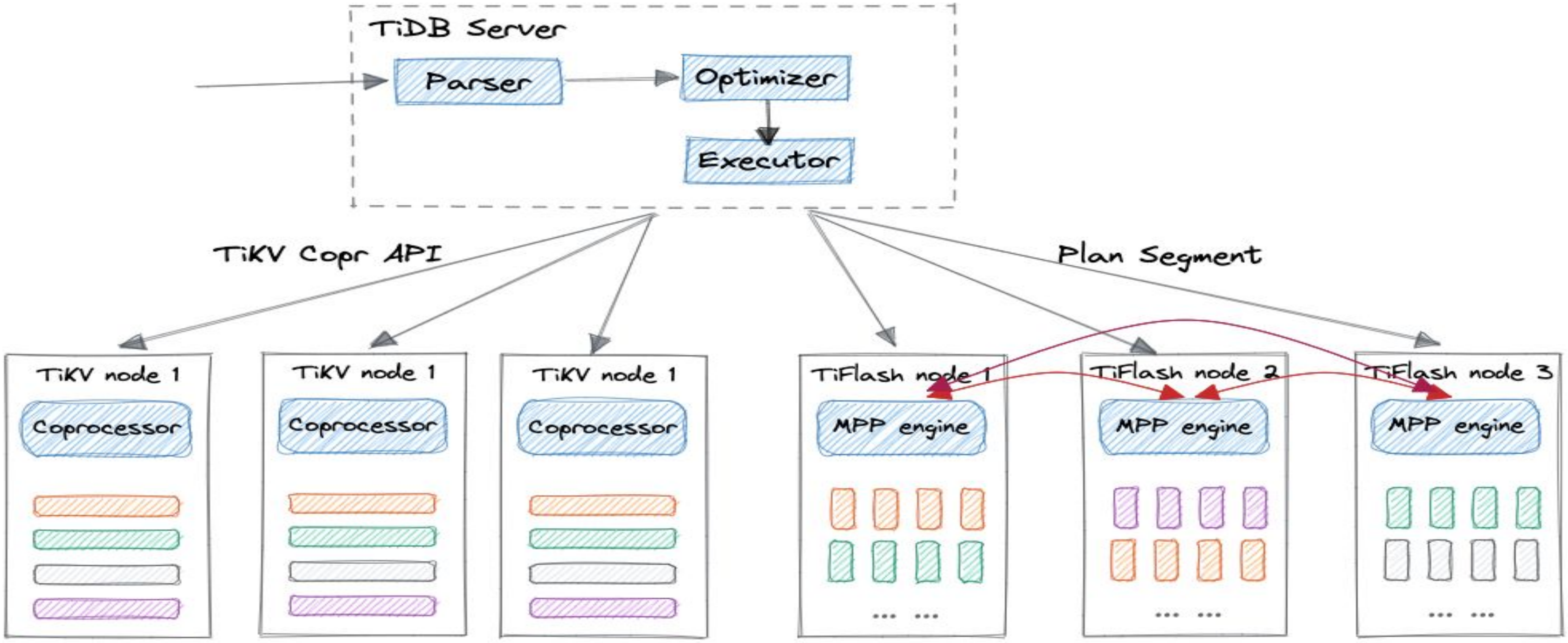
Optimizer Components

Generial Overview of the optimizer components



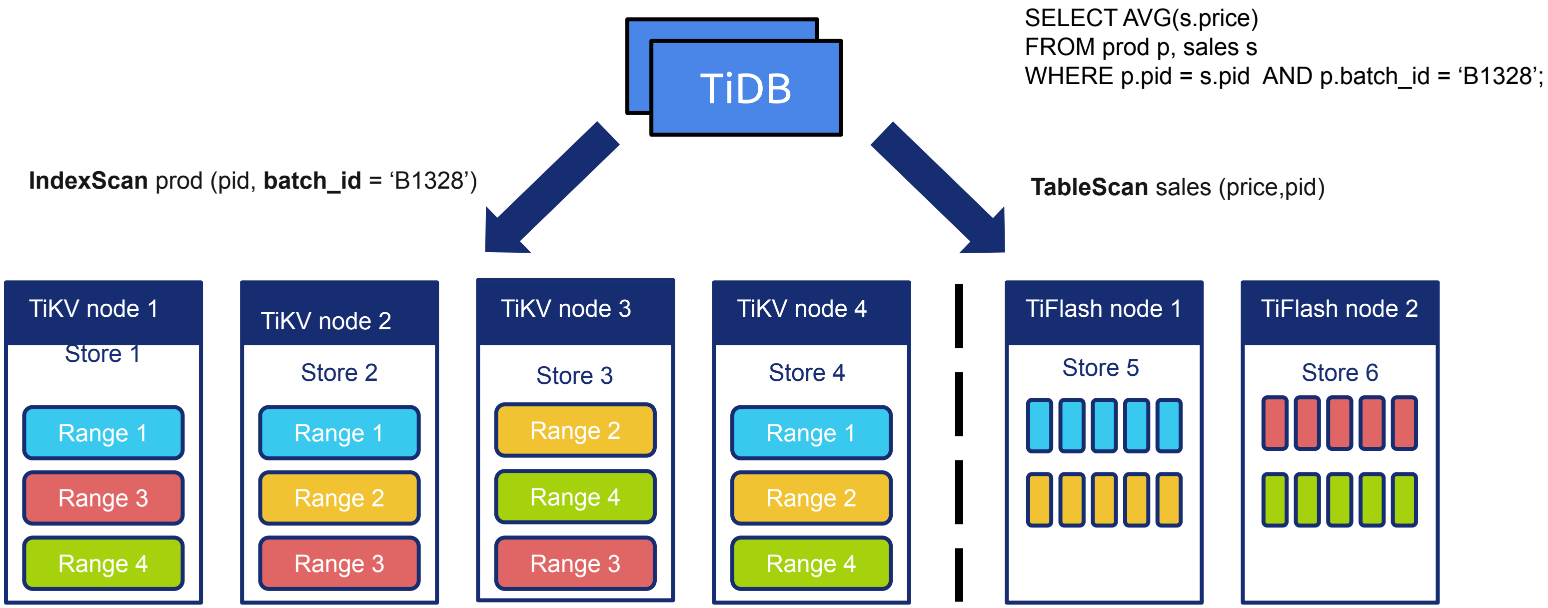
Bringing it all together

A brief look at how the SQL parser and optimizer work in TiDB



TiDB Optimizer

A brief look at how the SQL parser and optimizer work when [optional] TiFlash is installed



Agenda

01 Design Fundamentals

TiDB Architecture

02 Resource Control

Empowering Consolidated Workloads with Precision Resource Allocation. DXF.

03 Online DDL

Enhancing Database Agility with Lightning-Fast Schema Changes

04 Tools

TiDB's wide range of tools for managing your databases

Why TiDB Resource Control

Flow Control
resource quota

Resource segregation
CPU / IO

Fine-grained resource abstract
RU / RG

Usage tracking
Tune / Apportion

Schedule Control
job priority

When there are multiple apps/databases

Cost increase

Hard to maintain

Hard to cross database join

Consolidate?

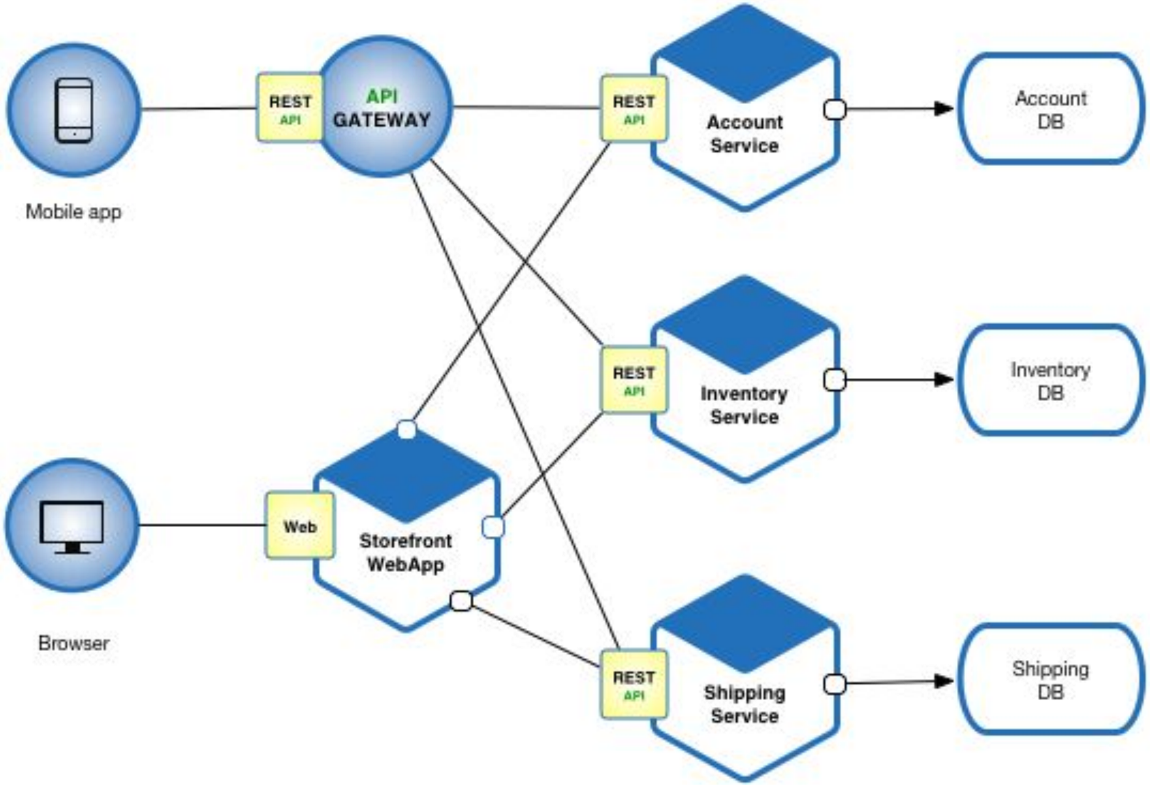
QoS

Change -> Disaster

Interfere

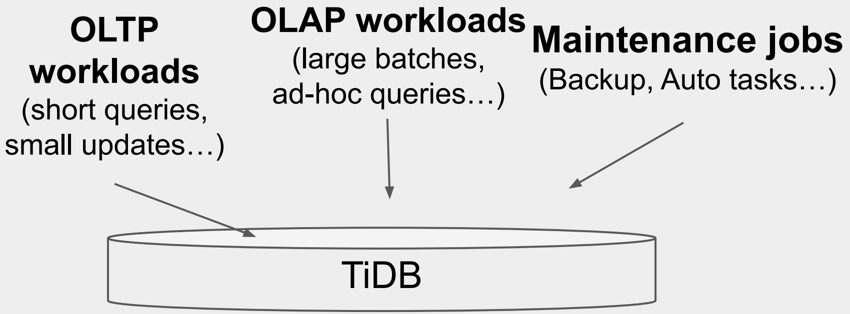
TiDB Resource Control

A typical microservice architecture: Database per service

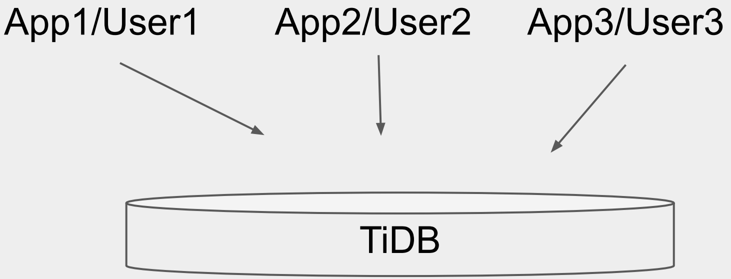


What is Resource Control?

Manage multiple workloads in a TiDB cluster.
Isolate, manage and schedule access to resources sharing the same TiDB cluster.



eg. xx workloads use too many resources and this impacts the P99 latency of small queries.



eg. Limit the resources allocated to app xxx/ user xxx.
eg. Allocate more resources to higher priority apps/users when the system is overloaded.

What is a Resource Group?

A resource group is a logical container for managing:

- **CPU**
- **I/O**

There are 3 important options for each resource group:

Option	Description
RU_PER_SEC	Rate of RU backfilling per second. Must be specified when creating a resource group.
PRIORITY	The absolute priority of tasks to be processed on TiKV. The default value is MEDIUM.
BURSTABLE	If the BURSTABLE attribute is set, use the available free system resources even if its quota is exceeded.

Request Unit (RU) and Scheduling

A Request Unit (RU) is an abstract unit for measuring system resource usage.

TiDB uses mClock, which is a **weight** and **constraint** based scheduler.

"...constraint-based scheduler ensures that [tasks] receive at least their minimum reserved service and no more than the upper limit in a time interval, while the weight-based scheduler allocates the remaining throughput to achieve proportional sharing."

Resource type	RU consumption
Read	2 storage read batches, 8 storage read requests and 64 KiB read request payload - consume 1 RU each
Write	1 storage write batch, 1 storage write request and 1 KiB write request - consume 1 RU each
SQL CPU	3 ms consumes 1 RU

Evaluate system capacity

- Estimate capacity based on hardware deployment and standard workloads

```
CALIBRATE RESOURCE;
+-------+
| QUOTA |
+-------+
| 190470 |
+-------+
1 row in set (0.01 sec)

CALIBRATE RESOURCE WORKLOAD
OLTP_WRITE_ONLY;
+-------+
| QUOTA |
+-------+
| 27444 |
+-------+
1 row in set (0.01 sec)
```

- Estimate capacity based on actual workloads

```
CALIBRATE RESOURCE START_TIME '2023-04-18
08:00:00' DURATION '20m';
+-------+
| QUOTA |
+-------+
| 27969 |
+-------+
1 row in set (0.01 sec)

CALIBRATE RESOURCE START_TIME '2023-04-18
08:00:00' END_TIME '2023-04-18 08:20:00';
+-------+
| QUOTA |
+-------+
| 27969 |
+-------+
1 row in set (0.01 sec)
```

Manage resource groups

Create Resource Group

```
CREATE RESOURCE GROUP IF NOT EXISTS rg1 RU_PER_SEC = 1000 BURSTABLE;
```

Alter Resource Group

```
ALTER RESOURCE GROUP rg1 RU_PER_SEC=20000 PRIORITY = HIGH;
```

Drop Resource Group

```
DROP RESOURCE GROUP rg1;
```

Query Resource Group(s)

```
SHOW CREATE RESOURCE GROUP rg1;
SELECT * FROM information_schema.resource_groups WHERE NAME = 'rg1';
```

Bind resource groups

User Level Mapping

```
CREATE USER 'user1'@'%' RESOURCE GROUP rg1;
ALTER USER 'user1' RESOURCE GROUP rg2;
SELECT User, User_attributes FROM mysql.user WHERE User = 'user1';
```

Session Level Mapping

```
SET RESOURCE GROUP <group name>
SELECT current_resource_group();
```

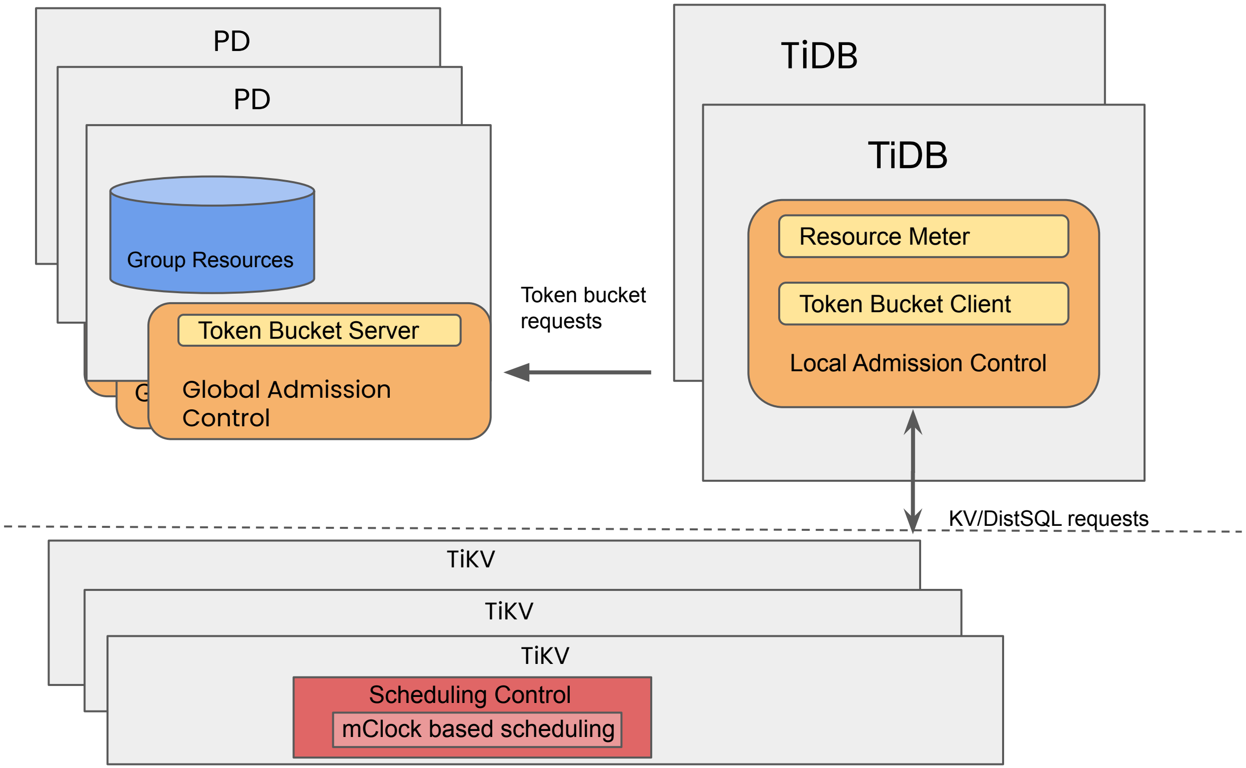
Statement Level Mapping

Hint: */*+ resource_group(${GROUP_NAME}) */*

```
SELECT /*+ resource_group(rg1) */ * FROM t1
INSERT /*+ resource_group(rg2) */ INTO t2 VALUES(2);
```

Statement (Hint) Level > Session Level > User Level

Resource Control Architecture



Admission Control Layer

- **Quota Limits by Request Unit**
- **GAC**
  - Maintain global token buckets
- **LAC**
  - Measure resources used by TiKV and TiDB (CPU + IO -> RU -> Tokens), consume tokens allocating by GAC

BURSTABLE

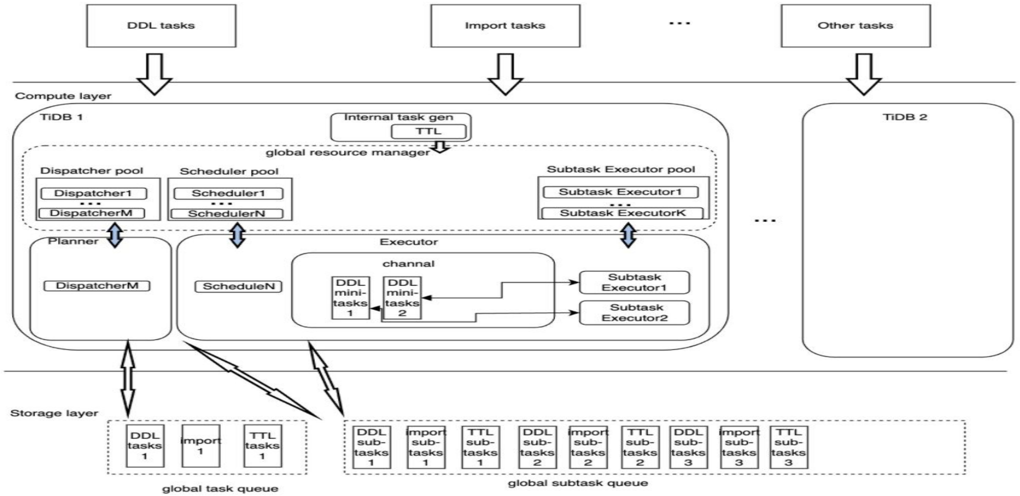
Scheduling Control Layer

- **Enhanced mClock based scheduling**
- **Weight input**
  - RU quota defined in resource groups
  - Priority defined in resource groups

Distributed eXecution Framework (DXF)

Apportion and control resources efficiently at the cluster level, to reduce impact on core business transactions

- Unified scheduling and distributed execution of tasks
- Unified resource management capabilities
- Provides **unified** capabilities for high scalability, high availability, and high performance
- Typical use cases: DDL, IMPORT, TTL, Analyze, Backup/Restore
    - Where a task processes large amount of data at both schema and table level
    - Executed periodically, but at a low frequency



Agenda

01 Design Fundamentals
TiDB Architecture

02 Resource Control
Empowering Consolidated Workloads with Precision Resource Allocation. DXF.

03 Online DDL
Enhancing Database Agility with Lightning-Fast Schema Changes

04 Tools
TiDB’s wide range of tools for managing your databases

MySQL solves DDL with MDL

MDL = Meta Data Lock

Table is locked for all sessions during the metadata (DD) update

ADD INDEX example, the metadata change **still needs to block**!

- MySQL uses a single instance/writer model
- Causes problems with MySQL replication
- Each MySQL replica will **asynchronously** run the DDL with an **MDL**
- Also if it's not an instant DDL, it makes the replication lag worse

Is a distributed database different?

Client connections see and act on the same data

Issues to solve (ADD INDEX as an example):

- No synchronous update of metadata/schemas for all cluster nodes
- Need to create index entries for all existing rows in the table
- Need to update entries for concurrent user changes

The Solution

Version all schemas.

Allow sessions to use current or the previous schema version

Use sub-state transitions:

- So that version N-1 is compatible with version N

Create states that will allow the full transition:

- From state ‘None/Start’ to state ‘Public’

	Public (vN)	**(vN-1)**			
SELECT	YES				
INSERT	YES				
UPDATE	YES				
DELETE	YES				

	Public (vN)	(vN-1)			
SELECT	YES	NO			
INSERT	YES	YES			
UPDATE	YES	YES			
DELETE	YES	YES			

	Public (vN+1)	Write Only (vN)	(vN-1)		
SELECT	YES	NO	NO		
INSERT	YES	YES	?		
UPDATE	YES	YES			
DELETE	YES	YES			

	Public (vN+1)	**Write Only (vN)**	**(vN-1)**		
SELECT	YES	NO	NO		
INSERT	YES	YES	NO - Backfill will handle it		
UPDATE	YES	YES			
DELETE	YES	YES			

	Public (vN+1)	Write Reorg (vN)	Write Only (vN-1)		
SELECT	YES	NO	NO	NO	
INSERT	YES	YES	YES	NO	
UPDATE	YES	YES	YES		
DELETE	YES	YES	YES		

	Public (vN+2)	Write Reorg (vN+1)	Write Only (vN)	(vN-1)	
SELECT	YES	NO	NO	NO	
INSERT	YES	YES	YES	NO	
UPDATE	YES	YES	YES	?	
DELETE	YES	YES	YES		

Index backfill

Table (Public)

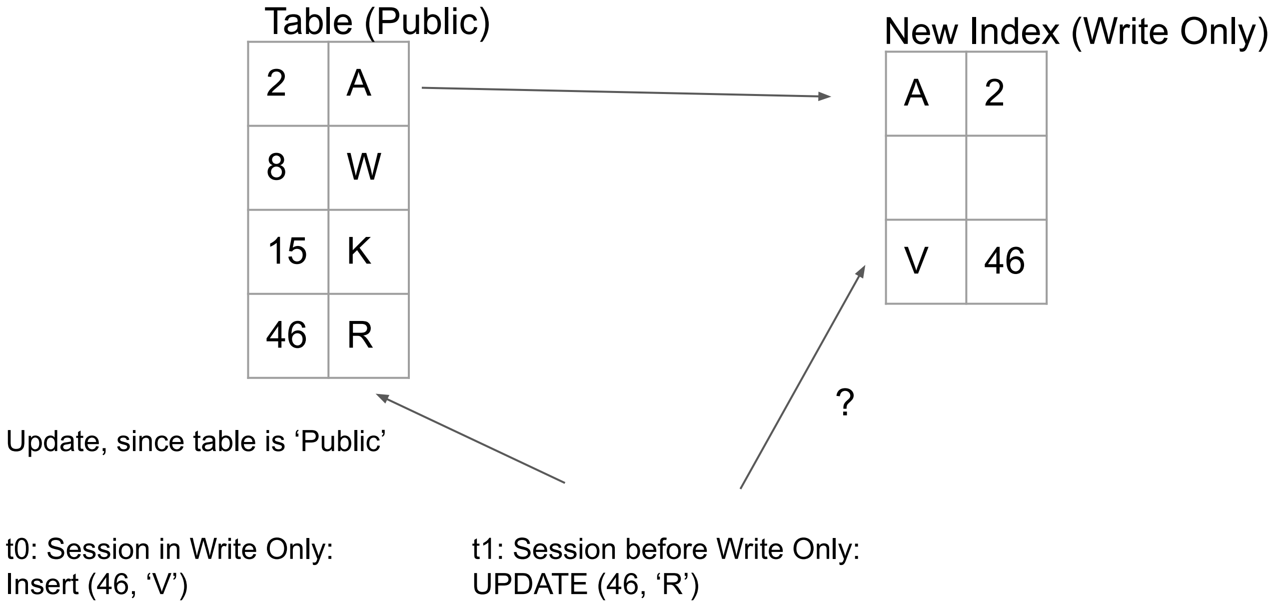
2	A
8	W
15	K
46	V

New Index (Write Only)

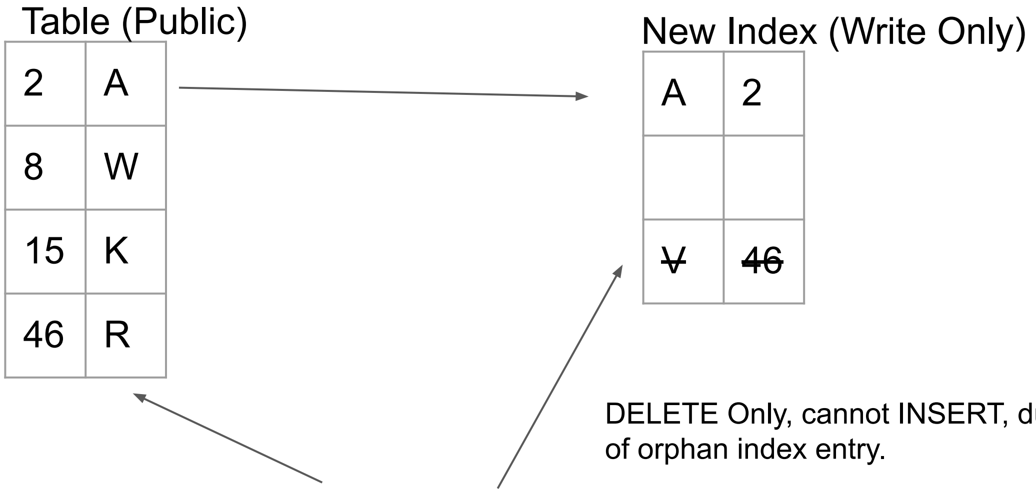
A	2
V	46

t0: Session in Write Only:
Insert (46, ‘V’)

Index backfill



Index backfill



DELETE Only, cannot INSERT, due to risk of orphan index entry.

t0: Session in Write Only:
Insert (46, ‘V’)

t1: Session before Write Only:
UPDATE (46, ‘R’)

	Public (vN+2)	Write Reorg (vN+1)	Write Only (vN)	(vN-1)	
SELECT	YES	NO	NO	NO	
INSERT	YES	YES	YES	NO	
UPDATE	YES	YES	YES	YES*	
DELETE	YES	YES	YES	?	

	Public (vN+2)	**Write Reorg (vN+1)**	**Write Only (vN)**	**Delete Only (vN-1)**	
SELECT	YES	NO	NO	NO	
INSERT	YES	YES	YES	NO	
UPDATE	YES	YES	YES	YES*	
DELETE	YES	YES	YES	YES	

	Public (vN+3)	Write Reorg (vN+2)	Write Only (vN+1)	Delete Only (vN)	None/Start (vN-1)
SELECT	YES	NO	NO	NO	NO
INSERT	YES	YES	YES	NO	NO
UPDATE	YES	YES	YES	YES*	NO
DELETE	YES	YES	YES	YES	NO

Other DDL Optimizations

RocksDB can ingest pre-generated SST files

We use pre-generated files for backfilling

- Generate SST files and ingest them into TiKV/RocksDB
- No need to write to the new index in TiKV
- Negligible impact on concurrent load
- Efficient use of network, CPU and IO

Use optimized Co-processor framework for reads

- Direct KV transactional reads are expensive
- Co-processor works on local data, avoids network overhead

ADD INDEX Timings

10 TiDB and 15 TiKV Nodes

Component	Hardware
TiDB	16 vCPU 32 GiB RAM - c6g.4 x large
PD	8 vCPU 16 GiB RAM - c6g.2 x large
TiKV	16 vCPU 64 GiB RAM 6T Disks - m6g.4 x large

Test	One-column Key Index	Ten-columns Key Index
10TB Table with Global Sort	47m	1 hour 6 min

Agenda

01 Design Fundamentals

TiDB Architecture

02 Resource Control

Empowering Consolidated Workloads with Precision Resource Allocation

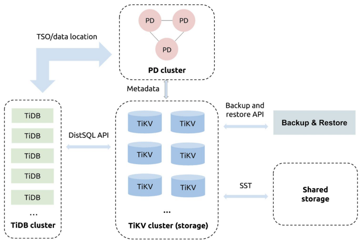
03 Online DDL

Enhancing Database Agility with Lightning-Fast Schema Changes

04 Tools

TiDB's wide range of tools for managing your databases

Tools - All Open Source



Backup/Restore

Dumpling

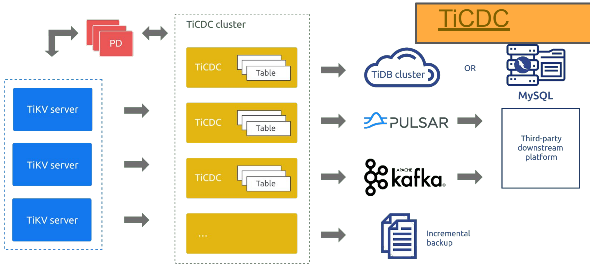
Export Tool

Lightning

Import Tool

Syncdiff

Comparison Tool



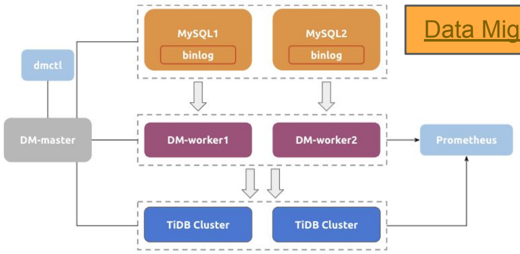
TiCDC

TiDB Operator

Automated operation and maintenance system for TiDB cluster in K8S

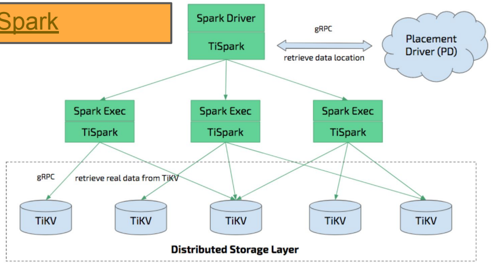
TiUP

Package manager for the TiDB ecosystem



Data Migration

TiSpark



Links

TiDB SQL Parser and Optimizer	TiDB’s SQL Parser and Optimizer
TiKV / Placement Driver	TiKV is a CNCF graduate project
TiFlash Column Store	TiDB’s column store engine for analytic queries
OSSInsight	GitHub realtime analytics with ~7 Billion GitHub events & growing
TiUP	Quick and easy way to try out TiDB
Join our Slack Channel	TiDB community slack channel
Chaos Mesh	Chaos engineering for Kubernetes

THANK YOU.