TiDB | COMMUNITY

Power TiKV with in-Memory Engine

{ Chenjie Tang PingCAP Storage Engineer }

TiDB | COMMUNITY

Catalogue

TiDB | COMMUNITY

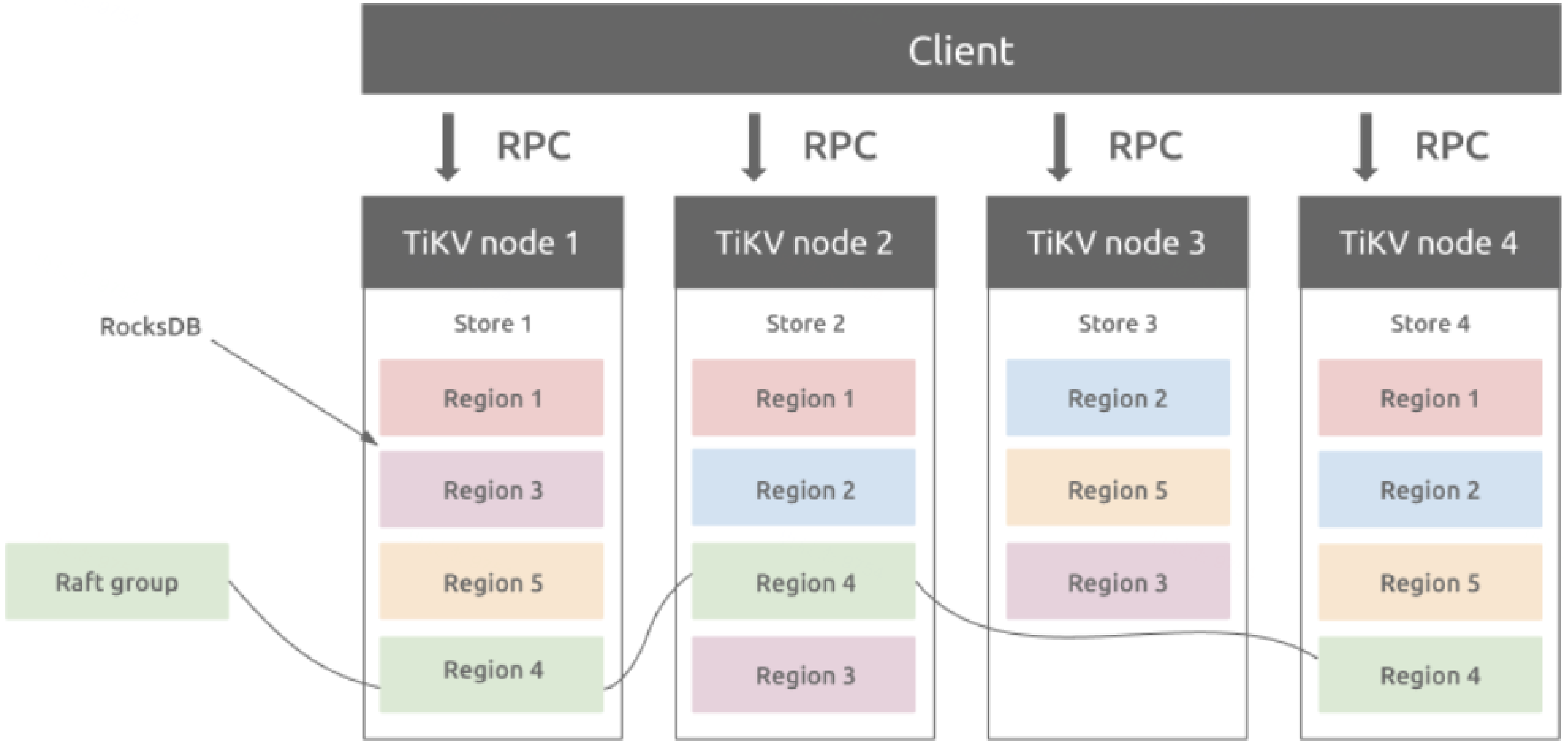
Part 1

TiKV Introduction

TiKV Introduction

TiDB | COMMUNITY

- Storage layer of TiDB
- Distributed key-value database
- Provides a distributed transaction (percolator) interface that satisfies ACID constraints
- Raft protocol guarantees multi-replication consistency and high availability



GitHub: https://github.com/tikv/tikv

Data in TiKV

TiDB | COMMUNITY

MVCC under TiKV:

```
key1-version4 -> value14
key1-version2 -> value12
key1-version1 -> value11
key2-version7 -> value27
key2-version3 -> value23
key2-version2 -> value22
key2-version1 -> value21
……
keyN-version2 -> valueN2
keyN-version1 -> valueN1
……
```

Data in TiKV

TiDB | COMMUNITY

Use version 4 to traverse:

```
key1-version4 -> value14
key1-version2 -> value12
key1-version1 -> value11
key2-version7 -> value27
key2-version3 -> value23
key2-version2 -> value22
key2-version1 -> value21
......
keyN-version2 -> valueN2
keyN-version1 -> valueN1
......
```

TiDB | COMMUNITY

Part 2

TiKV Pain Points in Read Scenario

Duplicate MVCC Versions Hurt Read Performance

Read two rows in the dataset:

```
key1-version100,
key1-version99,
..,
key1-version1,
key2-version100-delete,
key3-version100-delete,
..,
key10000-version100-delete,
key10001-version100
```

冗余 MVCC 版本影响读性能

TiDB | COMMUNITY

The metrics of real cluster（TPCC workload）：

Current Mitigations and Deficiencies

TiDB | COMMUNITY

Garbage Collection (GC)

- Compaction Filter
- Periodic GC

Problems

- Safe point block GC
- Compaction needs time

TiDB | COMMUNITY

Part 3

In-Memory Engine Overview

What is in-Memory Engine

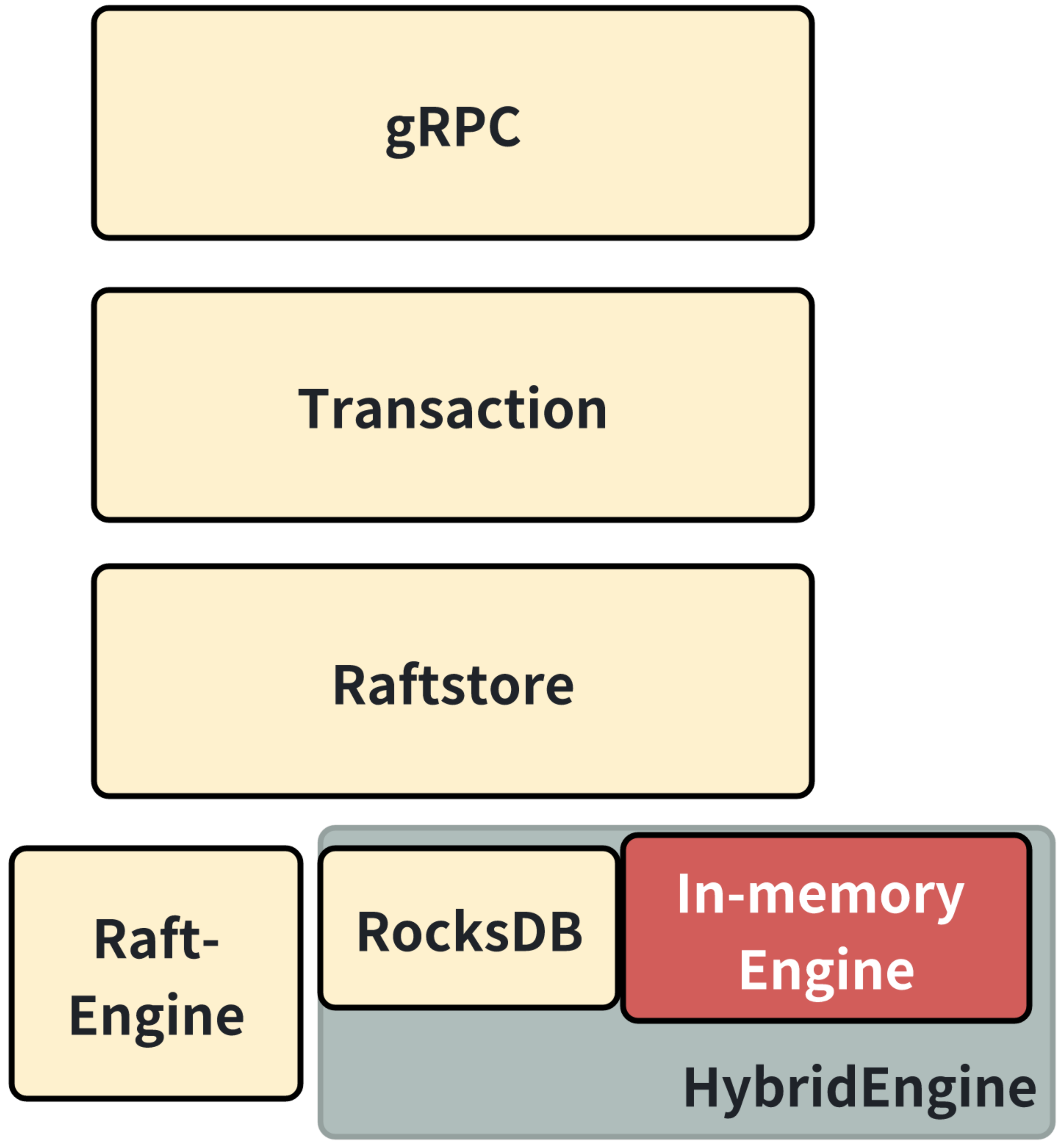
TiDB | COMMUNITY

- In-Memory cache
- Cache hot read regions
- Cached region holds “all” data
- Provide snapshot read
- Garbage collection

In-Memory Engine Architecture

TiDB | COMMUNITY

IME in TiKV



gRPC
- receive and handle clients' (ex TiDB) requets

Transaction
- handle distributed transactions
- send kv requests to Raftstore

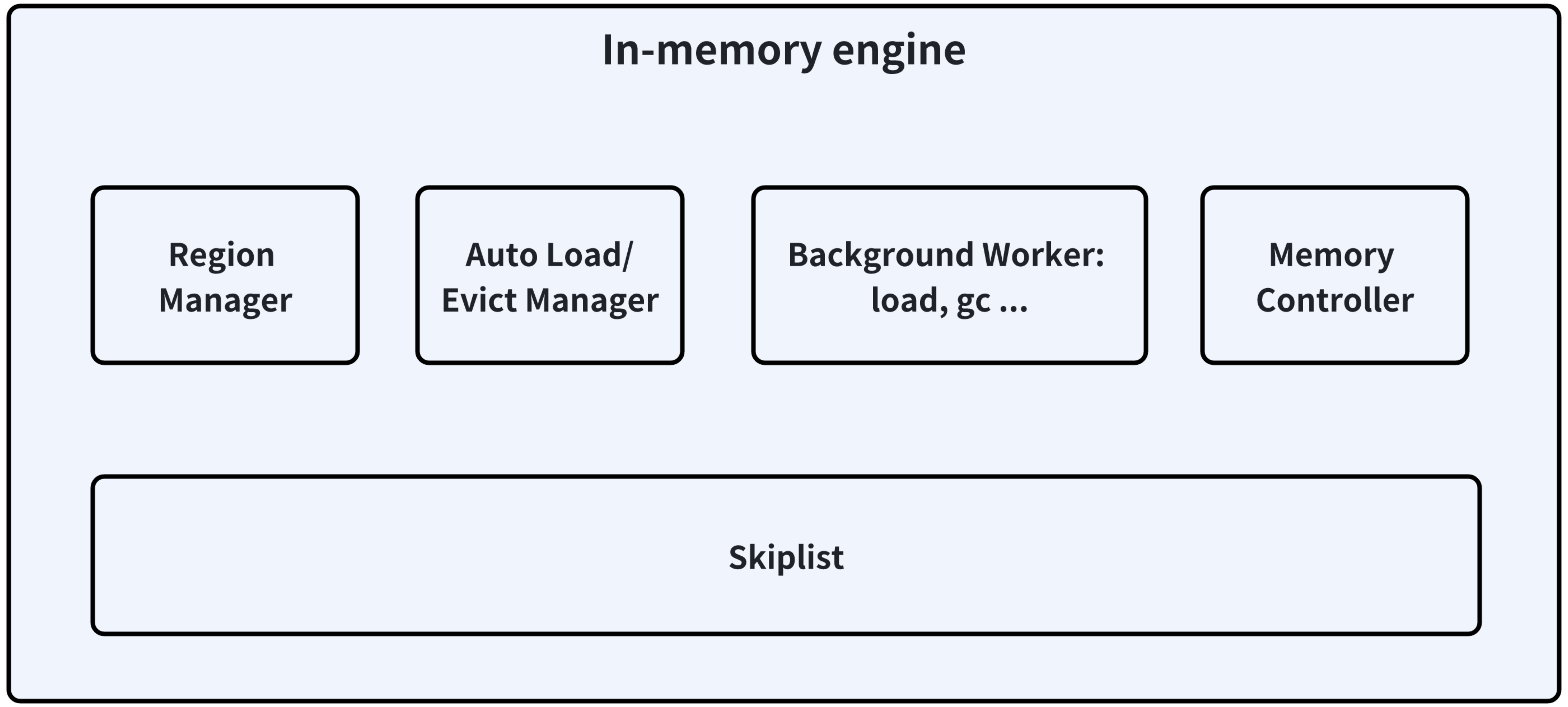
Raftstore
- Propose, replicate, commit, apply raft commands
- Region split and merge
- Region move
- …

Raft-Engine
- Persist raft commands

HybridEngine
- Persist KV

Important Modules in in-Memory Engine

TiDB | COMMUNITY



Skiplist
- store KV

Region Manager
- Manage region status
- Check validity to serve read request
- …

Auto Load/Evict Manager
- Which regions to load
- Which regions to evict

Background worker
- Execute load
- GC
- …

Memory Controller
- Manage memory usage

How to Solve Duplicate MVCC Version Problem

TiDB | COMMUNITY

Independent safe-point

- Enable to configure a shorter update period
- Will not be blocked by features such as flashback

Remove in-place when GC

- Skiplist can remove elements in-place

TiDB | COMMUNITY

Part 4

In-Memory Engine Design Detail

In-Memory Engine Design Detail

RFC: https://github.com/tikv/rfcs/pull/111

Tracking Issue: https://github.com/tikv/tikv/issues/16141

Read Flow

TiDB | COMMUNITY

Whether read request can be served:

- **Is region cached**
- **Is read ts larger than safe point**

Write Flow

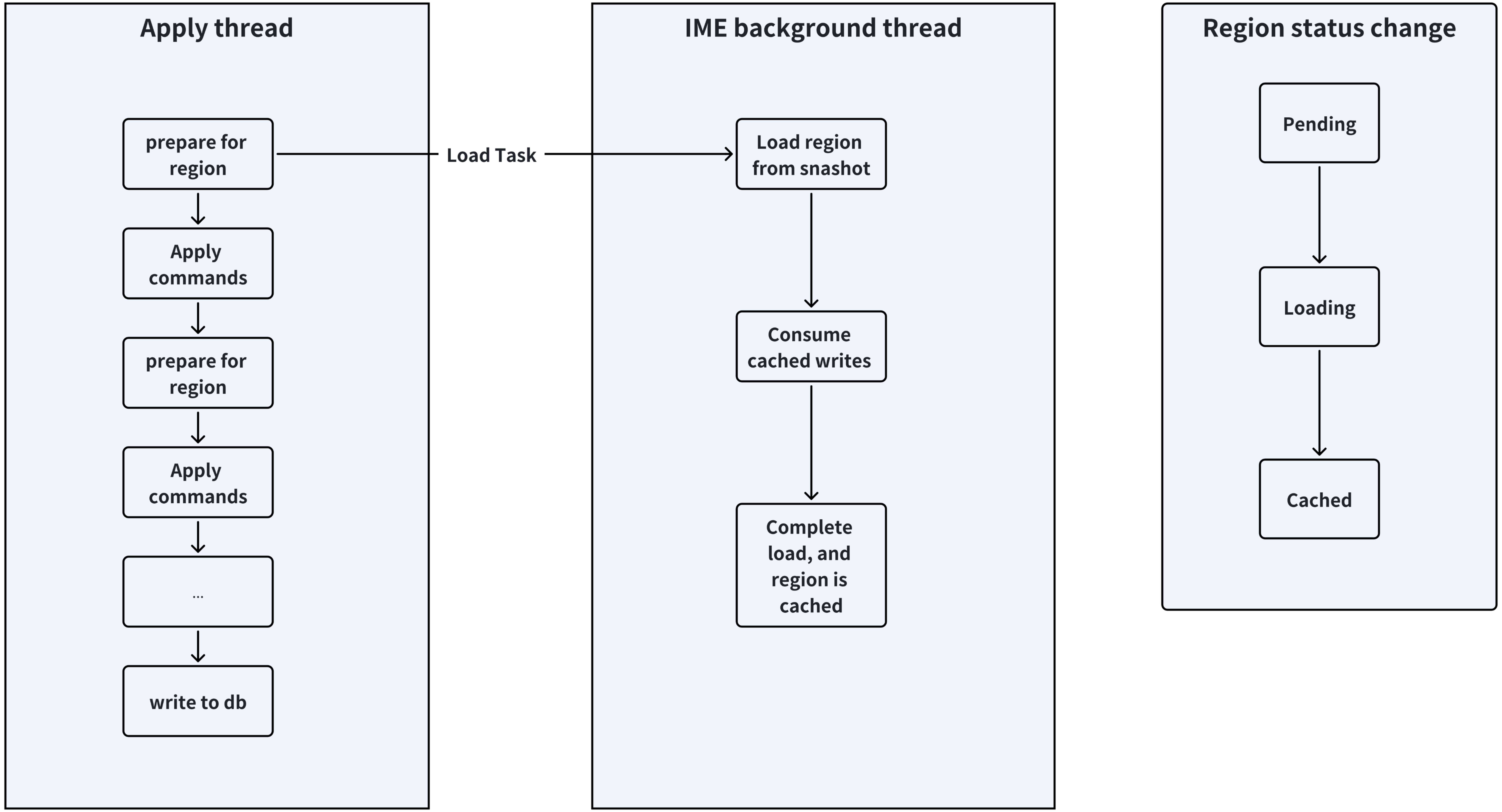
TiDB | COMMUNITY

Full write

Atomic Write

- RocksDB write batch atomicity
- Memtable callback

Load



Raft commands are executed in apply thread

Evict

TiDB | COMMUNITY

- Region cannot serve request after evicted
- Data can only be cleared after drop of all snapshots

TiDB | COMMUNITY

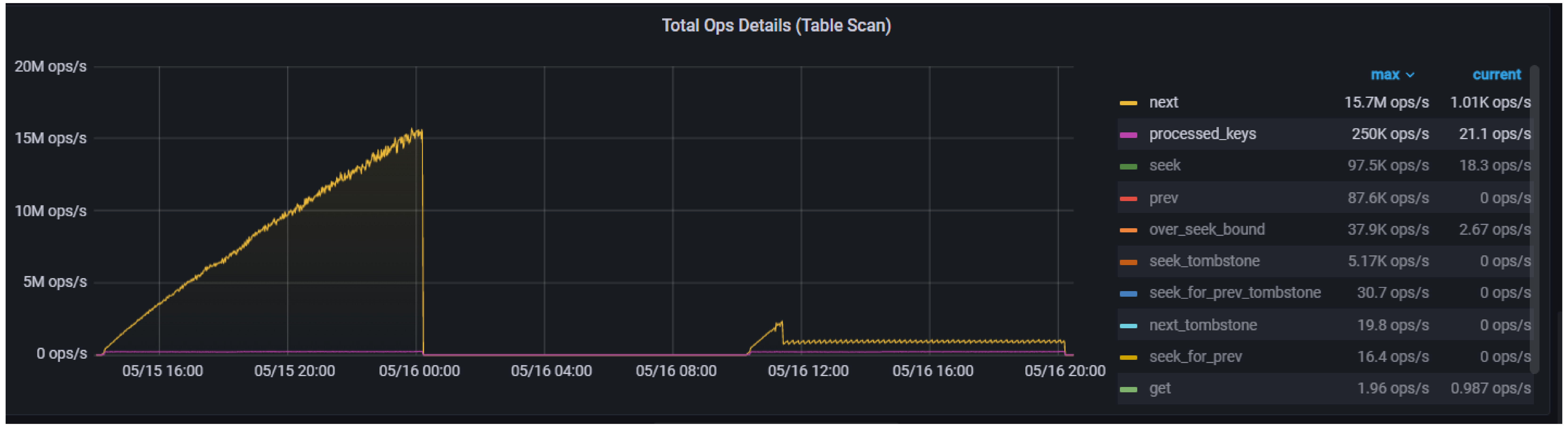
Part 5

In-Memory Engine Benchmark

TPCC benchmark

3 TiKV nodes, 16C CPU, 32GB RAM.
Run 10 hours
Safe point update duration 1 day

Tests	**Unified readpool CPU (per node)**	**Next /procseed_keys**	**Avg (coprocessor latency)**	**p80 (coprocessor latency)**	**p99 (coprocessor latency)**	**p999 (coprocessor latency)**
TPCC (without IME) 50 warehouses,	267%	15.5M ops/s / 250K ops/s	4ms	8.05ms	30.9ms	50ms
TPCC (with IME) 50 warehouses,	86%	900K ops/s / 249K ops/s	1.18ms	1.4ms	18ms	47.3ms



TPCC benchmark

TiDB | COMMUNITY

3 TiKV nodes, 16C CPU, 48GB RAM.
Safe point update duration 10 min (default)

Tests	Unified readpool CPU (per node)	Next /procseed_keys	Avg (coprocessor latency)	p80 (coprocessor latency)	p99 (coprocessor latency)	p999 (coprocessor latency)
TPCC (without IME) Small dataset, 50 warehouses, 11 GiB per tikv node	280%	8.35M ops/s / 350K ops/s	1.8ms	2.9ms	11.7ms	24.3ms
TPCC (with IME) Small dataset, 50 warehouses, 11 GiB per tikv node	230%	2M ops/s / 346K ops/s	1.3ms	1.5ms	11.2ms	24.7ms
TPCC (without IME) Large dataset, 1000 warehouses, 75 GiB per tikv node	430%	2M ops/s / 803K ops/s	4.5ms	3.0ms	50ms	150ms
TPCC (with IME) Large dataset, 1000 warehouses, 75 GiB per tikv node	400%	1.09M ops/s / 800K opsp/s	4.5ms	2.6ms	50ms	150ms

Note： the benchmark in this page used different clusters and parameters of client with the previous benchmark

TiDB | COMMUNITY

THANKS

PingCAP: https://www.pingcap.com/