PingCAP TiDB

Online schema change at scale in TiDB

Mattias Jonsson, PingCAP

MySQL solves DDL with MDL

MDL = Meta Data Lock
The table will be locked for all sessions while changing the metadata.

More and more operations will copy the data “online”, like ADD INDEX, but the metadata change **still needs to block**! But is also within a single instance.

In a replication chain, each replica will **asynchronousy** run the DLL with an MDL. Also if not instant DDL, it will cause replication delay.

Is a distributed database different?

In a distributed database like TiDB, all client connections sees and act on the same data. Just as expected from a transactional, ACID compliant, SQL database.

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.

How to solve it?

Proposed solution

- Version all schemas.
- Allow sessions to use current or the previous schema version.
- Use transitions, so that version N-1 is compatible with version N.

How can we 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			

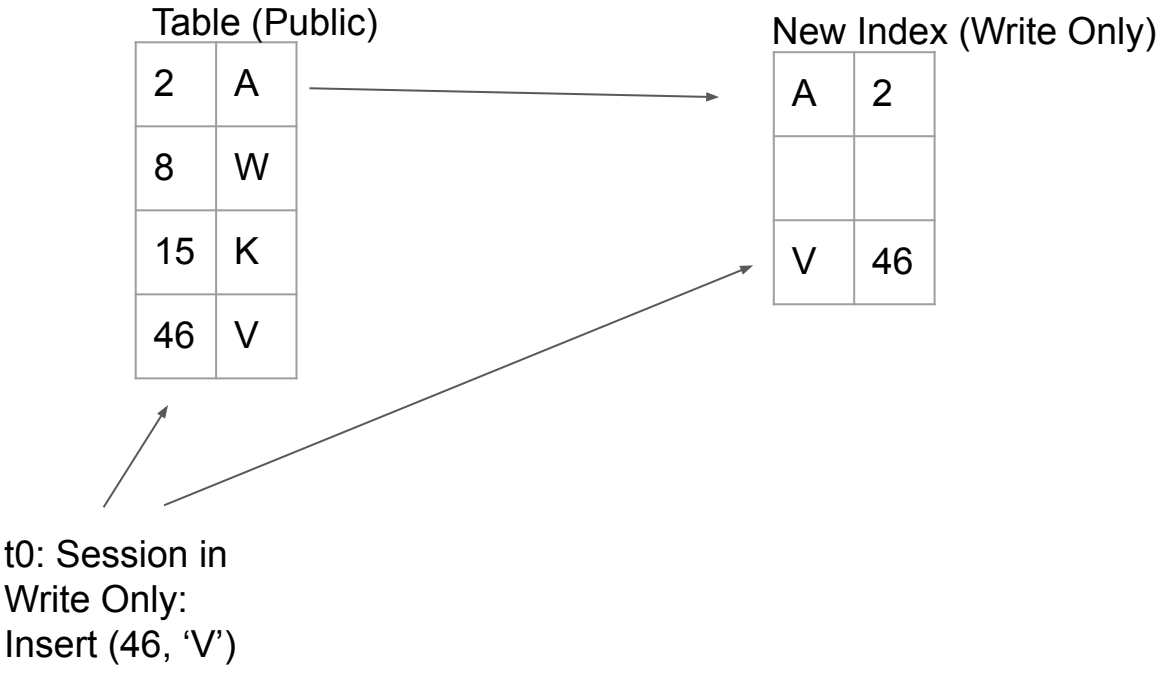
	Public (vN+1)	Write Only (vN)	(vN-1)		
SELECT	YES	NO	NO		
INSERT	YES	YES	?		
UPDATE	YES	YES			
DELETE	YES	YES			

	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			

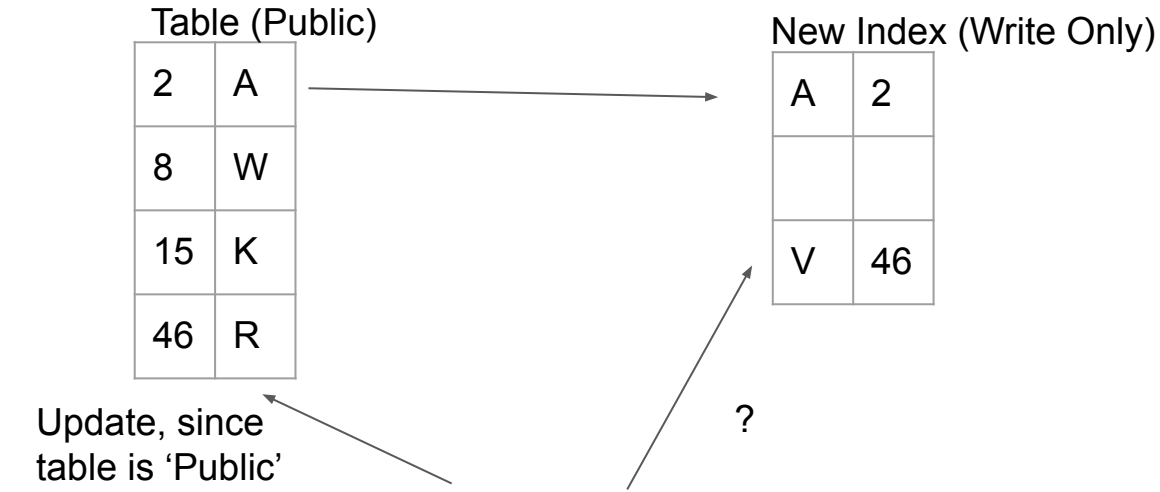
	Public (vN+1)	Write Reorg (vN)	Write Only (vN-1)		
SELECT	YES	NO	NO	NO	
INSERT	YES	YES	YES	NO	
UPDATE	YES	YES	YES		
DELETE	YES	YES	YES		

	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		

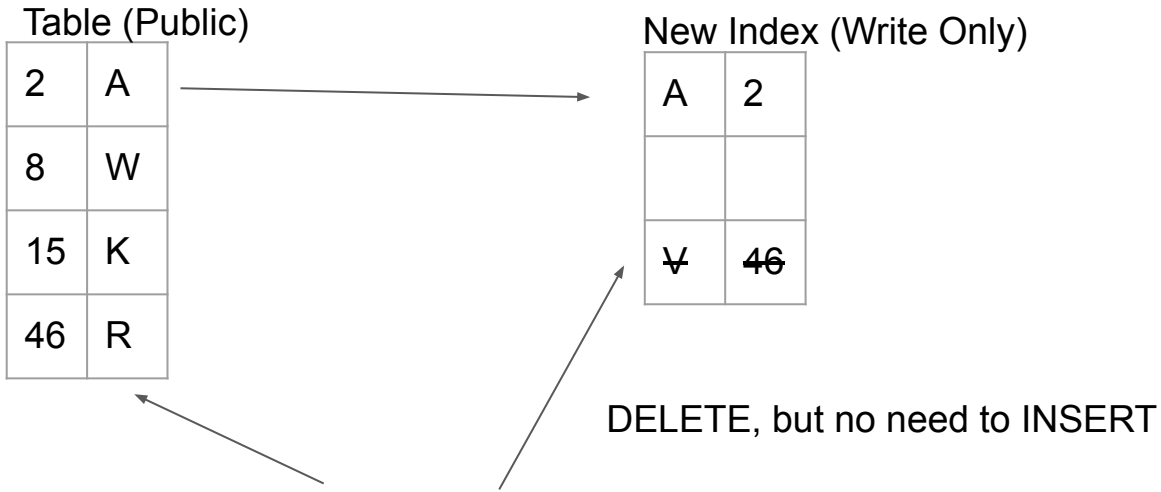
Index backfill



Index backfill



Index backfill



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	?	

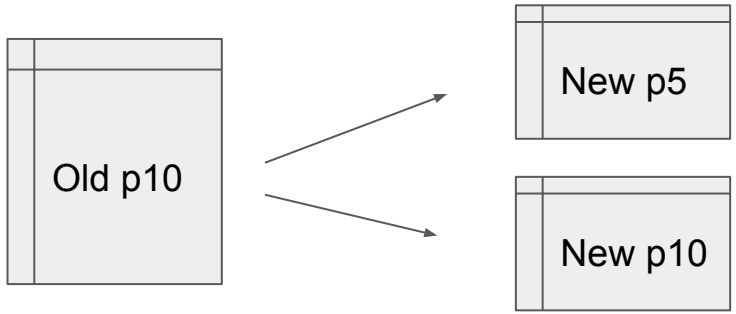
	Public (vN+2)	Write Reorg (vN+1)	Write Only (vN)	Delete Only (vN-1)	
SELECT	YES	NO	NO	NO	
INSERT	YES	YES	YES	NO	
UPDATE	YES	YES	YES	YES*	
DELETE	YES	YES	YES	YES	

	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

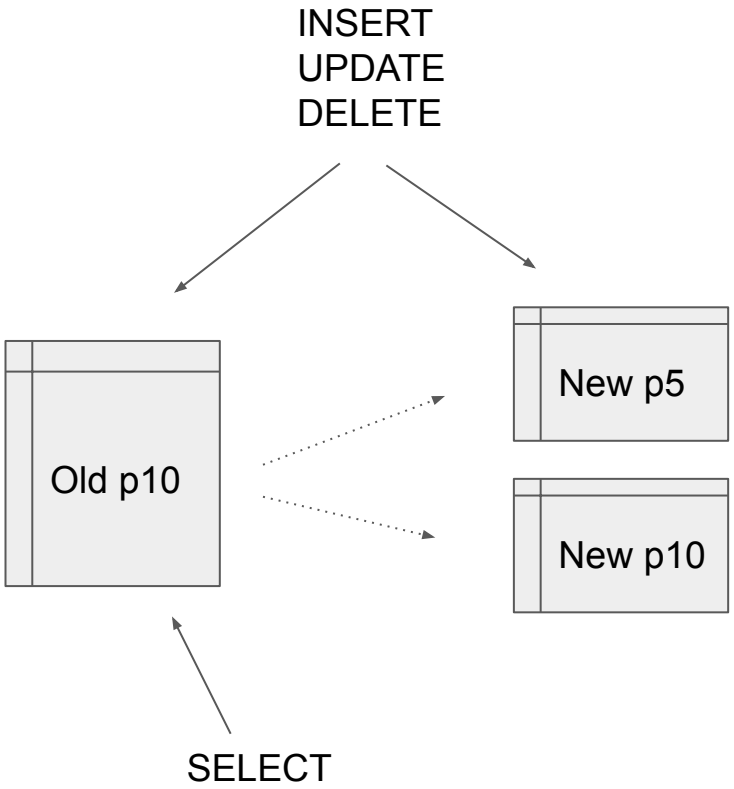
More complex case

ALTER TABLE t REORGANIZE PARTITION p10 INTO
(PARTITION p5 VALUES LESS THAN (5), PARTITION p10 VALUES LESS THAN (10))

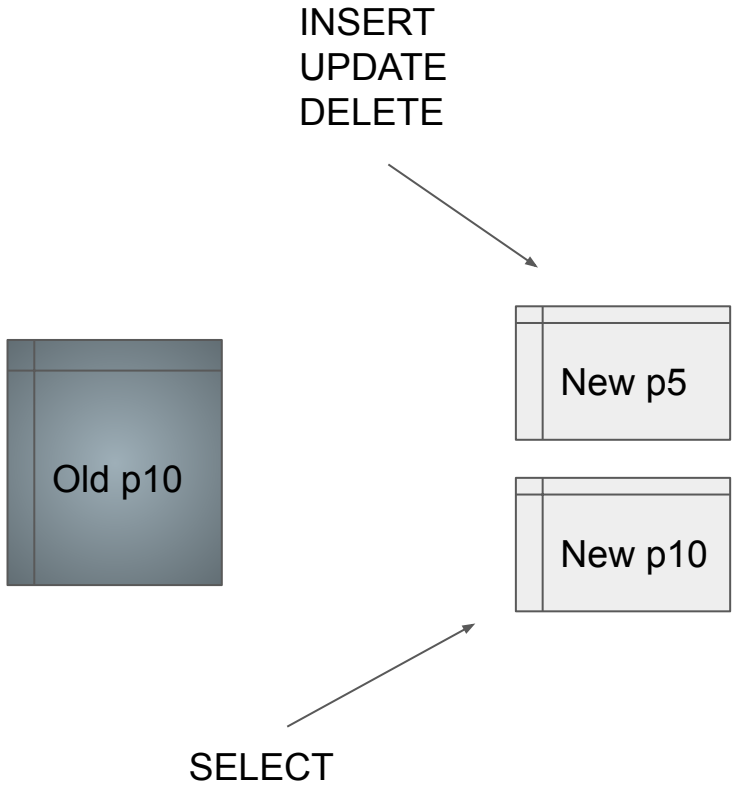
How to handle the partition swap after backfill?



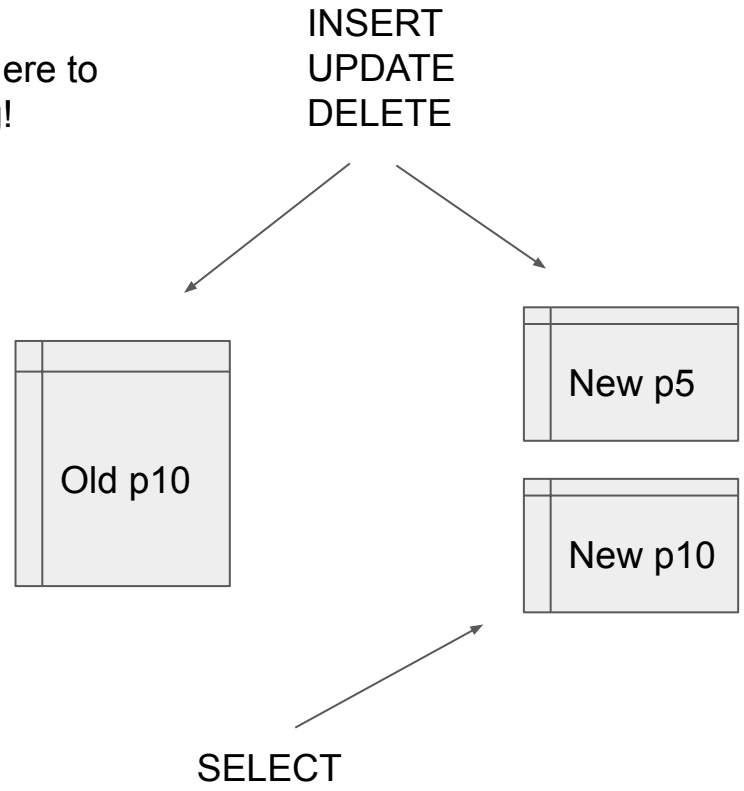
Write reorganization state



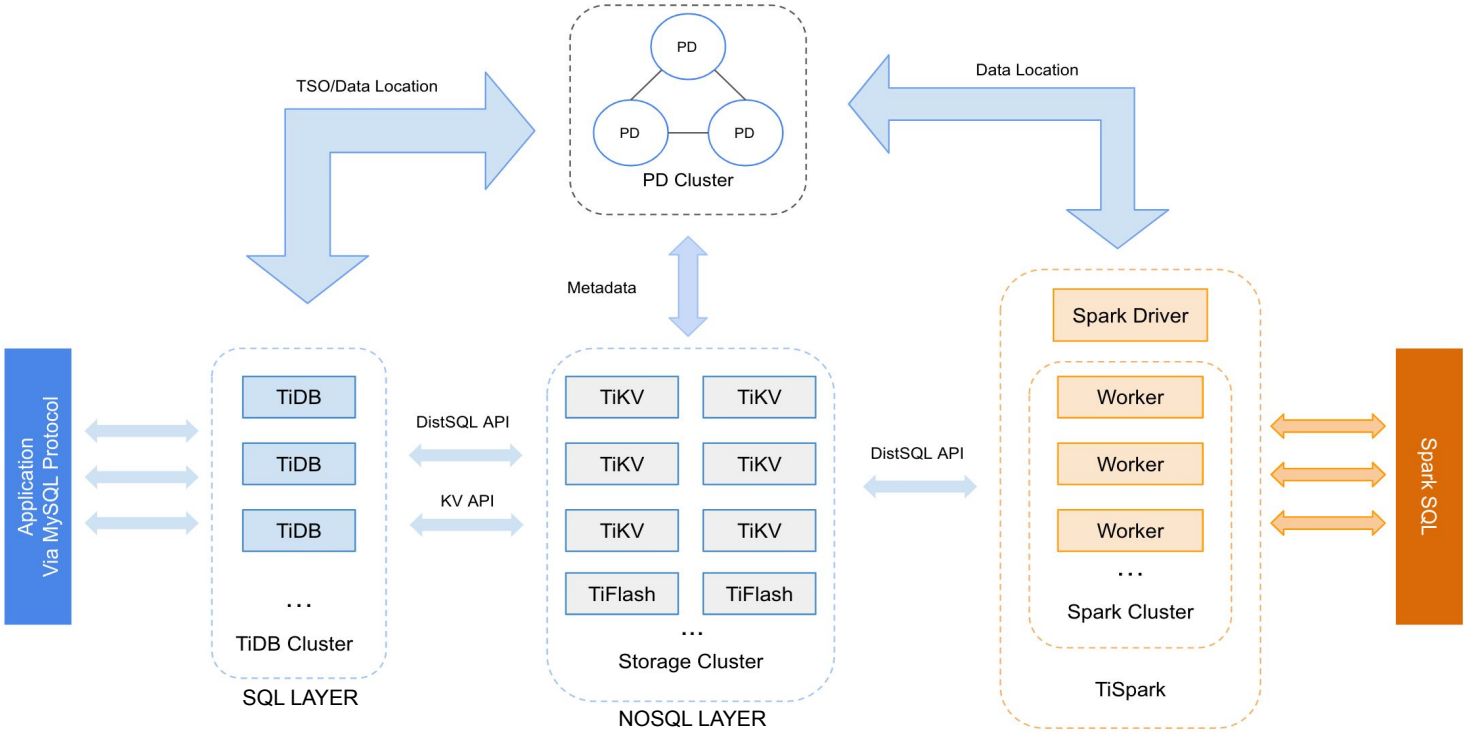
Public state



Additional state for changing where to read and keeping double writing!

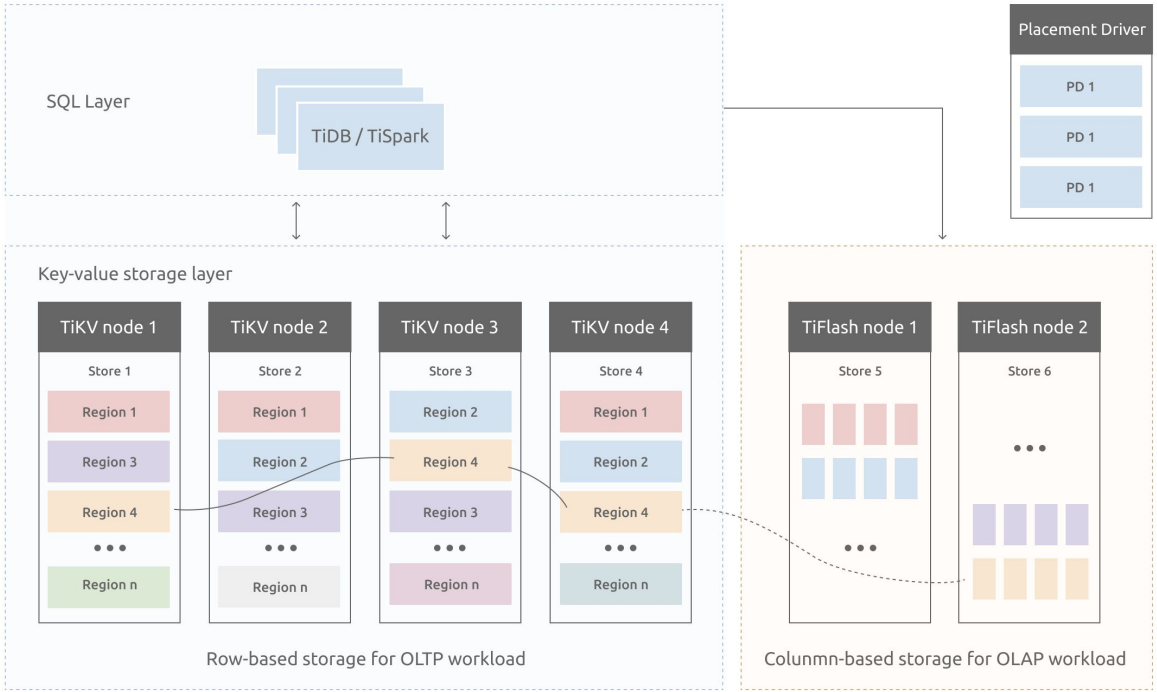


TiDB Architecture



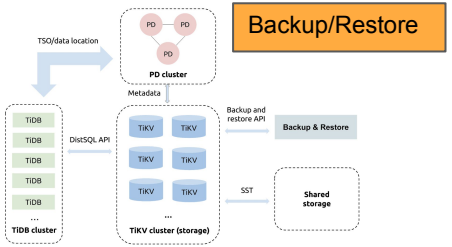
- Stateless SQL layer compatible with MySQL
- Distributed transactional key value storage for OLTP and Column storage for OLAP
- Apache Spark plug in

Raft based storage



- Separate storage for OLTP (row) and OLAP (columnar)
- Raft protocol for replication and distribution of data
- Data consistency
- Fault tolerance across Availability Zones

TiDB Tools



Backup/Restore

Dumpling

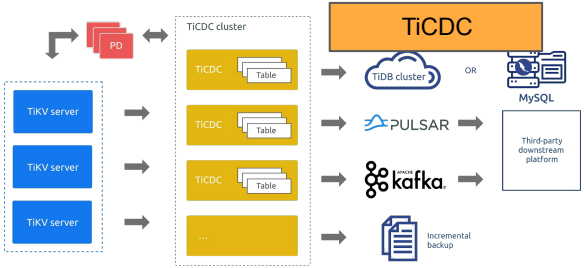
A Data Export Tool

Lightning

A Data Import Tool

Syncdiff

A Data Comparison Tool



TiCDC

TiDB Binlog

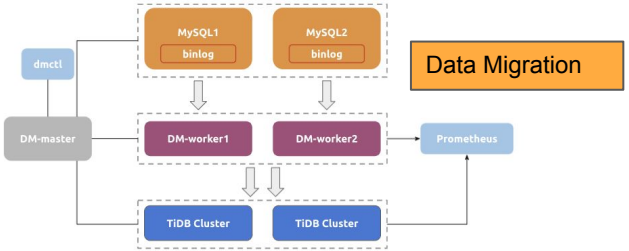
A Data Replication Tool

TiUP

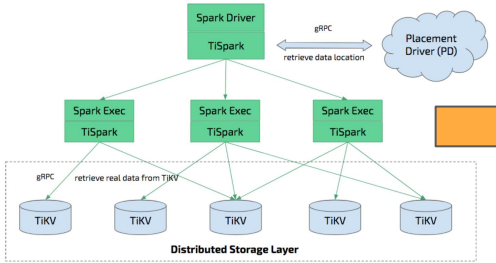
TiUP is a package manager to manage components of TiDB ecosystem

TiDB Operator

An automated operation and maintenance system for TiDB cluster in K8S



Data Migration

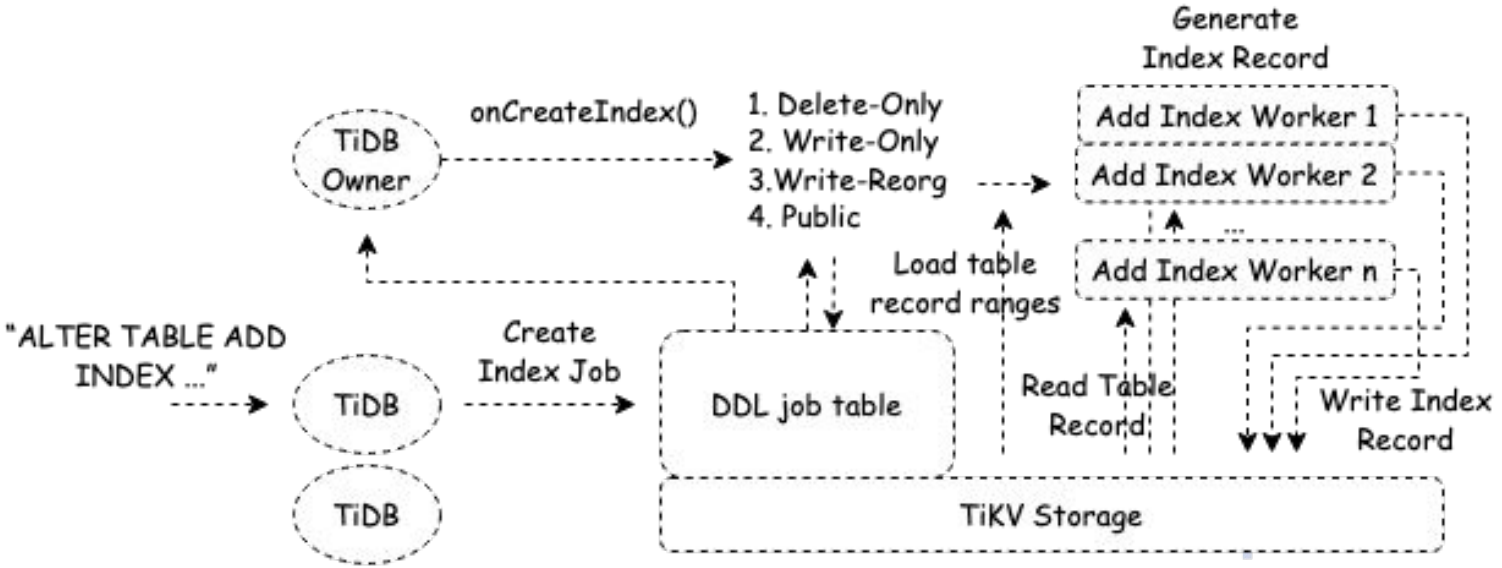


TiSpark

Faster ADD INDEX in TiDB

- State transitions proven and stable.
- Similar products where faster (~3x).
- Non optimized implementation.
- Data copy/index build was done in small transaction batches.
- Only a single node as DDL owner/executer

Overview of ADD INDEX



tidb_enable_fast_ddl

- Writing entries in transactional batches is expensive and slow.
- RocksDB can ingest pre-generated SST files.
- In v6.3 we added a new way of backfilling, instead of writing to a new index in TiKV, generate SST files and ingest them into TiKV/RocksDB
- Result is ~ 3X speedup.
- And a lot less impact on concurrent load (less network, cpu and IO)

Tracking issue: github.com/pingcap/tidb/issues/35983

Further optimizations

- Less wait between batches, better scheduling
- Use optimized Co-processor framework for reads instead of direct KV transactional reads
- Disconnect Read->Write dependency, make it asynchronous.
- + other smaller optimizations
- Results in 3X - 5X speedup

10x improvement since v6.1 LTS release to v6.5 LTS release**!**

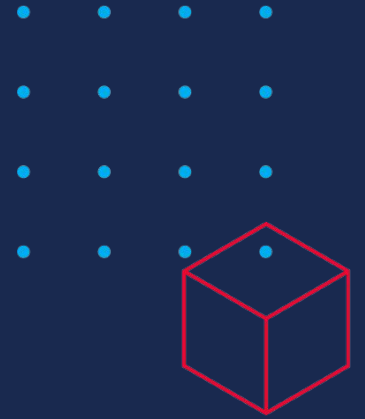
Future optimizations

- Still only using a single TiDB node executing the DDL, we are currently working on how to distribute the work if resources are available.
- Auto tune priority between production load and DDL.

Links

- github.com/pingcap/tidb
- github.com/tikv/tikv / github.com/tikv/pd
  (TiKV is a Cloud Native Computing Foundation graduate project)
- github.com/pingcap/tiflash
- OSSInsight.io Analytics/Demo site, with 5.5+ Billion github events in a single table.
- tiup.io for simple deploy/testing
- slack.tidb.io TiDB Community slack channel
- github.com/chaos-mesh Chaos Engineering for Kubernetes

PingCAP TiDB



Thanks