TiDB 3.0: What's new and what's next?

PingCAP TiDB

Ed Huang, CTO @ PingCAP
h@pingcap.com

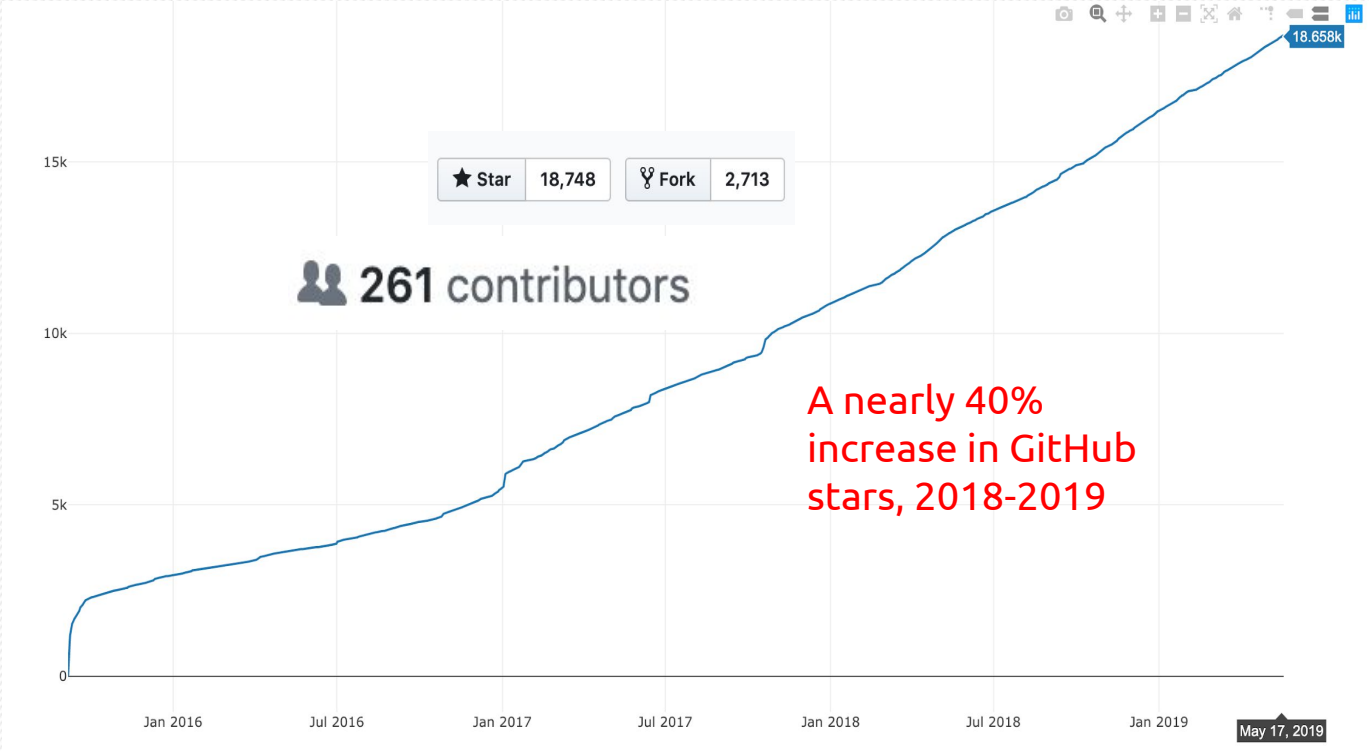
What's TiDB?

- Elastic scaling (scaling is transparent to the application layer)
- Speaks MySQL dialect with ACID semantics
- High availability with auto-failover
- Open source
- Goal: A real HTAP database

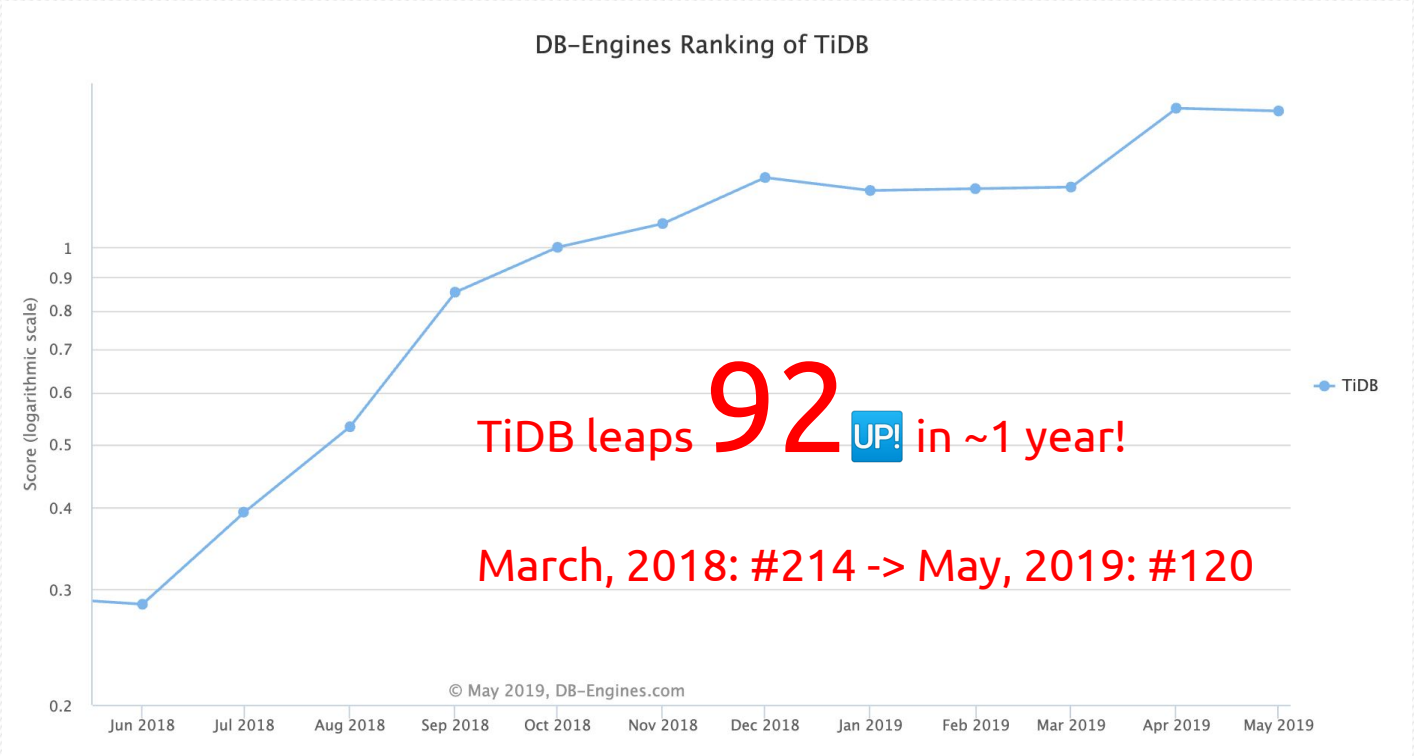
Not a fork, not middleware or storage engine.

“Several TB”

GitHub star trend of TiDB



TiDB is climbing fast in DB-Engines ranking!



Who's using TiDB?

Who's using TiDB？

uses TiDB to serve its core banking system

has 30 TiDB clusters with nearly 200 physical nodes

has 20 TiDB clusters with nearly 32 TB of data

has > 5 TiDB clusters with nearly 80 TB of data

What our users say

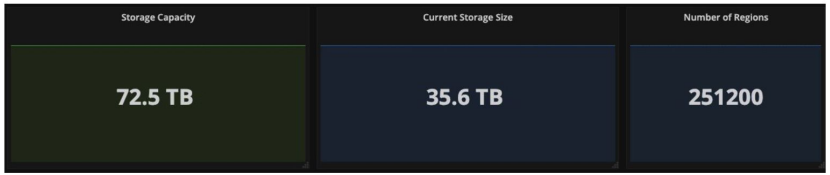
Blitzscaling the Largest Dockless Bikesharing Platform with TiDB's Help

It has been one year since we deployed TiDB in our production environment. In the past year, the number of our users has increased nearly ten times and the daily riding data has grown dozens of times. Thanks to the online scalability of TiDB, we have successfully scaled our infrastructure. We can finally focus on the development and optimization of Mobike applications to deliver amazing experiences for our user, without worrying about sharding rules for MySQL. This is extremely valuable for a fast-growing startup, like us, giving us a head-start in a competitive environment.

Shopping on Shopee, the TiDB Way

Thu, Feb 14, 2019 Chunhui Liu, Chao Hong

So far, our system has been running smoothly with the data volume growing to 35TB at the time of this writing (See Figure 5). After scaling our capacity twice, there are now 42 nodes in the cluster.

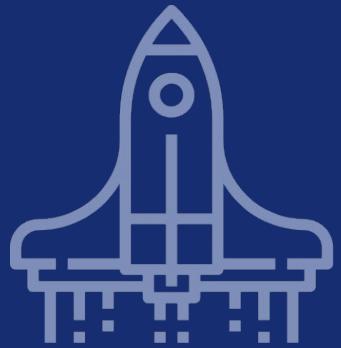


BookMyShow.com: More Uptime, 30% Less Operational Cost with TiDB

> ***"Operational and maintenance cost has been reduced by 30%. No engineer needs to be fully dedicated to database operations anymore."***

PingCAP

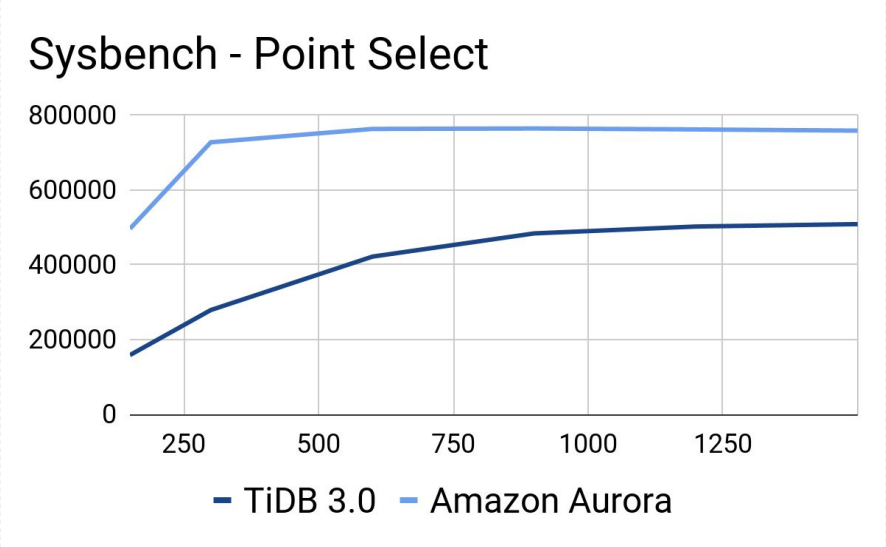
What's new in TiDB 3.0

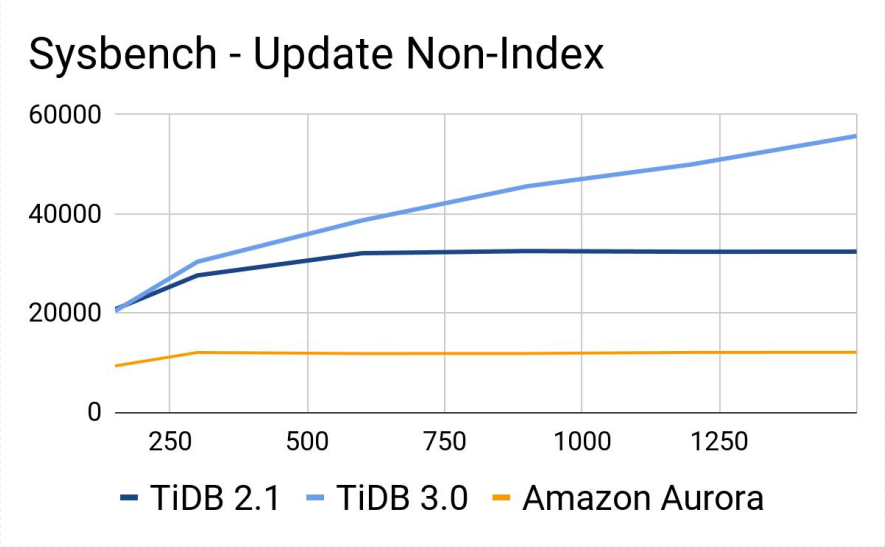


Improved Performance with Select and Update

All tests are done in AWS with three c5d.4xlarge for tidb-server, three c5d.4xlarge for tikv-server.

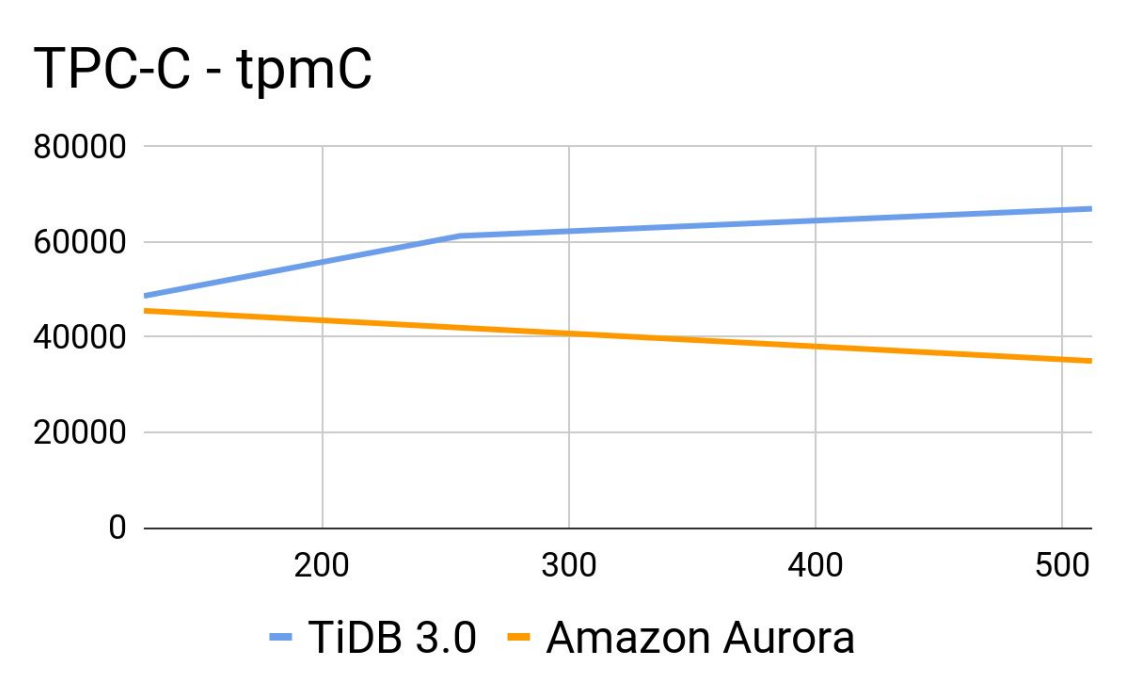
500K reads/second and 55K writes/second on a 3 node TiDB cluster!





Measured in Throughput. Higher is better

More Complex, Decision Support Queries (TPC-C Benchmark)



TiDB + TiKV: 6 x c5d.4xlarge (16 vCPU, 32 GB mem)
Aurora: 6 x db.r4.4xlarge (16 vCPU, 122 GB mem)

both TiDB and Aurora are deployed in one region

Measured in Throughput. Higher is better

Role-Based Access Control (RBAC)

Example:

```
CREATE DATABASE newdb;
CREATE ROLE 'app_developer';
GRANT ALL ON newdb.* TO 'app_developer';
CREATE USER 'dev';
GRANT 'app_developer' TO 'dev';
SET DEFAULT ROLE app_developer TO 'dev';
```

Example: `dev` user can only handle the authorized DB

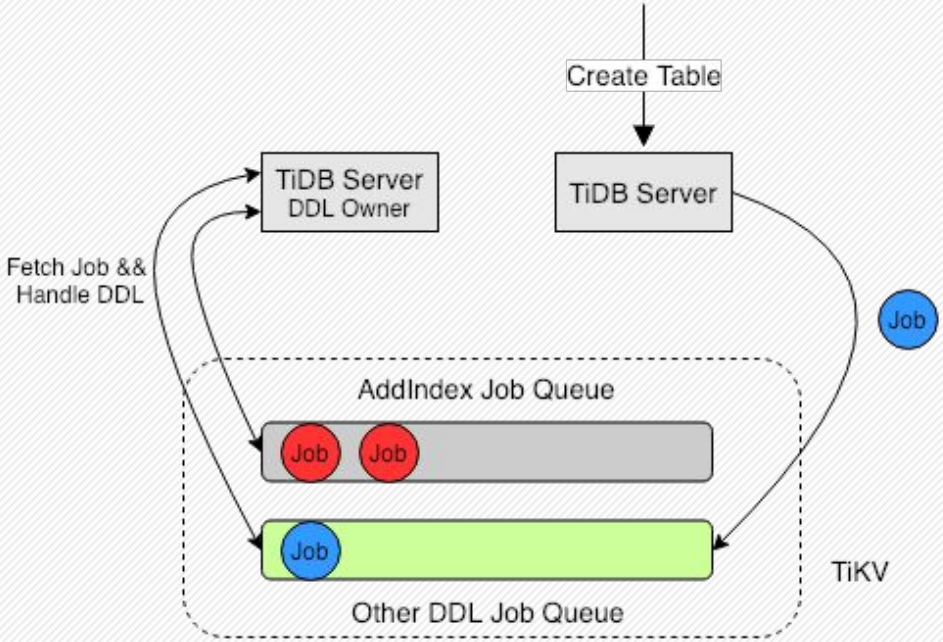
```
master* $ mysql --host 127.0.0.1 -P 4001 -u dev

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.25-TiDB-v3.0.0-rc.1-90-gf6346a1e8 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)]> show databases;
+--------------------+
| Database           |
+--------------------+
| INFORMATION_SCHEMA |
| newdb              |
+--------------------+
2 rows in set (0.000 sec)
```

```
master* $ mysql --host 127.0.0.1 -P 4001 -u root

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.25-TiDB-v3.0.0-rc.1-90-gf6346a1e8 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)]> show databases;
+--------------------+
| Database           |
+--------------------+
| INFORMATION_SCHEMA |
| PERFORMANCE_SCHEMA |
| mysql              |
| newdb              |
| test               |
+--------------------+
5 rows in set (0.000 sec)
```

PingCAP

Online DDL

- Non-blocking!
- Modifications that only involve meta-information of the table are completed immediately
  - Add column
  - Remove column
  - Create table
  - Drop table
- Support for MySQL DDL assertions:
  - ALGORITHM=INSTANT
  - ALGORITHM=INPLACE
- TiDB never uses ALGORITHM=COPY or LOCK=SHARED|EXCLUSIVE



Range and hash partitioning

- No subpartitioning
- Logical partition
- Data placement hint
- Help optimizer generate a more efficient query plan.

Example:

```
CREATE TABLE t (
        a INT(10) UNSIGNED NOT NULL,
        b DATETIME NOT NULL,
        PRIMARY KEY (a, b)
) PARTITION BY RANGE (TO_DAYS(b))
(PARTITION p20190101 VALUES LESS THAN (TO_DAYS('2019-01-01')),
 PARTITION p20190201 VALUES LESS THAN (TO_DAYS('2019-02-01')),
 PARTITION p20190301 VALUES LESS THAN (TO_DAYS('2019-03-01')),
 PARTITION p20190401 VALUES LESS THAN (TO_DAYS('2019-04-01')),
 PARTITION p00000000 VALUES LESS THAN MAXVALUE);
```

Example: Optimizer generates a more efficient query plan

```
MySQL [test]> EXPLAIN SELECT * FROM t WHERE b < CAST('2019-04-03' AS DATETIME);
+-------------------------+---------+------+-------------------------------------------------------------------------------+
| id                      | count   | task | operator info                                                                 |
+-------------------------+---------+------+-------------------------------------------------------------------------------+
| Union_11                | 6650.99 | root |                                                                               |
| ├─TableReader_14        | 2.99    | root | data:Selection_13                                                             |
| │ └─Selection_13        | 2.99    | cop  | lt(test.t.b, 2019-04-03 00:00:00)                                             |
| │   └─TableScan_12      | 3.00    | cop  | table:t, partition:p20190101, range:[-inf,+inf], keep order:false, stats:pseudo |
| ├─TableReader_17        | 0.33    | root | data:Selection_16                                                             |
| │ └─Selection_16        | 0.33    | cop  | lt(test.t.b, 2019-04-03 00:00:00)                                             |
| │   └─TableScan_15      | 3.00    | cop  | table:t, partition:p20190201, range:[-inf,+inf], keep order:false, stats:pseudo |
| ├─TableReader_20        | 3323.33 | root | data:Selection_19                                                             |
| │ └─Selection_19        | 3323.33 | cop  | lt(test.t.b, 2019-04-03 00:00:00)                                             |
| │   └─TableScan_18      | 3.00    | cop  | table:t, partition:p20190301, range:[-inf,+inf], keep order:false, stats:pseudo |
| ├─TableReader_23        | 3323.33 | root | data:Selection_22                                                             |
| │ └─Selection_22        | 3323.33 | cop  | lt(test.t.b, 2019-04-03 00:00:00)                                             |
| │   └─TableScan_21      | 3.00    | cop  | table:t, partition:p20190401, range:[-inf,+inf], keep order:false, stats:pseudo |
| └─TableReader_26        | 1.00    | root | data:Selection_25                                                             |
|   └─Selection_25        | 1.00    | cop  | lt(test.t.b, 2019-04-03 00:00:00)                                             |
|     └─TableScan_24      | 3.00    | cop  | table:t, partition:p00000000, range:[-inf,+inf], keep order:false, stats:pseudo |
+-------------------------+---------+------+-------------------------------------------------------------------------------+
16 rows in set (0.001 sec)

MySQL [test]> EXPLAIN SELECT * FROM t WHERE b < CAST('2019-01-03' AS DATETIME);
+-------------------------+-------+------+-------------------------------------------------------------------------------+
| id                      | count | task | operator info                                                                 |
+-------------------------+-------+------+-------------------------------------------------------------------------------+
| Union_8                 | 3.32  | root |                                                                               |
| ├─TableReader_11        | 2.99  | root | data:Selection_10                                                             |
| │ └─Selection_10        | 2.99  | cop  | lt(test.t.b, 2019-01-03 00:00:00)                                             |
| │   └─TableScan_9       | 1.00  | cop  | table:t, partition:p20190101, range:[-inf,+inf], keep order:false, stats:pseudo |
| └─TableReader_14        | 0.33  | root | data:Selection_13                                                             |
|   └─Selection_13        | 0.33  | cop  | lt(test.t.b, 2019-01-03 00:00:00)                                             |
|     └─TableScan_12      | 1.00  | cop  | table:t, partition:p20190201, range:[-inf,+inf], keep order:false, stats:pseudo |
+-------------------------+-------+------+-------------------------------------------------------------------------------+
7 rows in set (0.001 sec)
```

Information schema

- Observing internal status is now more DBA friendly!
- Several new INFORMATION_SCHEMA tables:
  - Slow query (SLOW_QUERY)
  - Hot data range (TIDB_HOT_REGIONS)
  - Storage nodes status / Data distribution status (TIKV_STORE_STATUS)

```
MySQL [INFORMATION_SCHEMA]> show tables;
+---------------------------------------+
| Tables_in_INFORMATION_SCHEMA          |
+---------------------------------------+
| ANALYZE_STATUS                        |
| CHARACTER_SETS                        |
| …                                     |
| SCHEMA_PRIVILEGES                     |
| SESSION_STATUS                        |
| SESSION_VARIABLES                     |
| SLOW_QUERY                            |
| …                                     |
| TIDB_HOT_REGIONS                      |
| TIDB_INDEXES                          |
| TIKV_REGION_PEERS                     |
| TIKV_REGION_STATUS                    |
| TIKV_STORE_STATUS                     |
| TRIGGERS                              |
| USER_PRIVILEGES                       |
| VIEWS                                 |
+---------------------------------------+
```

Information schema

Example: Show the three slowest queries

```
MySQL [(none)]> select sleep(10);
+-----------+
| sleep(10) |
+-----------+
|         0 |
+-----------+
1 row in set (10.002 sec)

MySQL [(none)]> select sleep(10);
+-----------+
| sleep(10) |
+-----------+
|         0 |
+-----------+
1 row in set (10.003 sec)

MySQL [(none)]> select sleep(15);
+-----------+
| sleep(15) |
+-----------+
|         0 |
+-----------+
1 row in set (15.002 sec)
```

```
MySQL [INFORMATION_SCHEMA]> desc INFORMATION_SCHEMA.SLOW_QUERY;
+--------------+--------------------+------+------+---------+-------+
| Field        | Type               | Null | Key  | Default | Extra |
+--------------+--------------------+------+------+---------+-------+
| Time         | timestamp unsigned | YES  |      | NULL    |       |
| Txn_start_ts | bigint(20) unsigned | YES  |      | NULL    |       |
| User         | varchar(64)        | YES  |      | NULL    |       |
| Conn_ID      | bigint(20) unsigned | YES  |      | NULL    |       |
| Query_time   | double unsigned    | YES  |      | NULL    |       |
| Process_time | double unsigned    | YES  |      | NULL    |       |
| Wait_time    | double unsigned    | YES  |      | NULL    |       |
| Backoff_time | double unsigned    | YES  |      | NULL    |       |
| Request_count | bigint(20) unsigned | YES  |      | NULL    |       |
| Total_keys   | bigint(20) unsigned | YES  |      | NULL    |       |
| Process_keys | bigint(20) unsigned | YES  |      | NULL    |       |
| DB           | varchar(64)        | YES  |      | NULL    |       |
| Index_ids    | varchar(100)       | YES  |      | NULL    |       |
| Is_internal  | tinyint(1) unsigned | YES  |      | NULL    |       |
| Digest       | varchar(64)        | YES  |      | NULL    |       |
| Stats        | varchar(512)       | YES  |      | NULL    |       |
| Cop_proc_avg | double unsigned    | YES  |      | NULL    |       |
| Cop_proc_p90 | double unsigned    | YES  |      | NULL    |       |
| Cop_proc_max | double unsigned    | YES  |      | NULL    |       |
| Cop_proc_addr | varchar(64)        | YES  |      | NULL    |       |
| Cop_wait_avg | double unsigned    | YES  |      | NULL    |       |
| Cop_wait_p90 | double unsigned    | YES  |      | NULL    |       |
| Cop_wait_max | double unsigned    | YES  |      | NULL    |       |
| Cop_wait_addr | varchar(64)        | YES  |      | NULL    |       |
| Mem_max      | bigint(20) unsigned | YES  |      | NULL    |       |
| Query        | varchar(4096)      | YES  |      | NULL    |       |
+--------------+--------------------+------+------+---------+-------+
26 rows in set (0.000 sec)
```

```
MySQL [INFORMATION_SCHEMA]> select Query, Query_time
    -> from INFORMATION_SCHEMA.SLOW_QUERY
    -> ORDER BY Time DESC LIMIT 3;
+--------------------+--------------+
| Query              | Query_time   |
+--------------------+--------------+
| select sleep(15); | 15.001227832 |
| select sleep(10); | 10.002010125 |
| select sleep(10); | 10.001084817 |
+--------------------+--------------+
3 rows in set (0.002 sec)
```

PingCAP

Information schema

Example: Show storage nodes' status

```
MySQL [INFORMATION_SCHEMA]> select STORE_ID,ADDRESS,VERSION,CAPACITY,LEADER_COUNT,REGION_SCORE,UPTIME
    -> FROM TIKV_STORE_STATUS
    -> ORDER BY STORE_ID ASC;
+----------+-----------------+-------------+----------+--------------+--------------+--------------+
| STORE_ID | ADDRESS         | VERSION     | CAPACITY | LEADER_COUNT | REGION_SCORE | UPTIME       |
+----------+-----------------+-------------+----------+--------------+--------------+--------------+
|        1 | 127.0.0.1:20168 | 3.0.0-beta.1 | 932 GiB  |            2 |           14 | 10m0.257224s |
|        2 | 127.0.0.1:20165 | 3.0.0-beta.1 | 932 GiB  |            3 |            5 | 10m1.150844s |
|        7 | 127.0.0.1:20169 | 3.0.0-beta.1 | 932 GiB  |            0 |            4 | 10m1.169562s |
|        8 | 127.0.0.1:20160 | 3.0.0-beta.1 | 932 GiB  |            2 |            5 | 10m1.133782s |
|        9 | 127.0.0.1:20161 | 3.0.0-beta.1 | 932 GiB  |            1 |            4 | 10m0.202262s |
|       10 | 127.0.0.1:20163 | 3.0.0-beta.1 | 932 GiB  |            4 |            5 | 10m1.150705s |
|       11 | 127.0.0.1:20162 | 3.0.0-beta.1 | 932 GiB  |            2 |            4 | 10m0.169546s |
|       12 | 127.0.0.1:20166 | 3.0.0-beta.1 | 932 GiB  |            1 |            4 | 10m1.150789s |
|       13 | 127.0.0.1:20164 | 3.0.0-beta.1 | 932 GiB  |            1 |            4 | 10m1.190715s |
|       14 | 127.0.0.1:20167 | 3.0.0-beta.1 | 932 GiB  |            2 |            5 | 10m0.219701s |
+----------+-----------------+-------------+----------+--------------+--------------+--------------+
10 rows in set (0.002 sec)
```

Optimizer improvements

- Statistics
  - Kept more up-to-date with incremental analysis
  - Faster to generate with intelligent sampling
- More optimal plan generation
  - Improvements to cost model
  - Skyline pruning
  - Join re-order
- Improved Observability
  - EXPLAIN ANALYZE support
  - Query tracing

Starting from TiDB 3.0, TiDB optimizer has been able to provide the best query plan for all TPC-H Queries.

EXPLAIN ANALYZE

- As an extension to EXPLAIN, **EXPLAIN ANALYZE** executes the query and provide additional execution statistics
  - **time**: Elapsed time for this operator
  - **loop**: The number of times the operator was called from the parent operator
  - **rows**: The number of rows that were returned by this operator

EXPLAIN ANALYZE

Example: TPCH-Q17

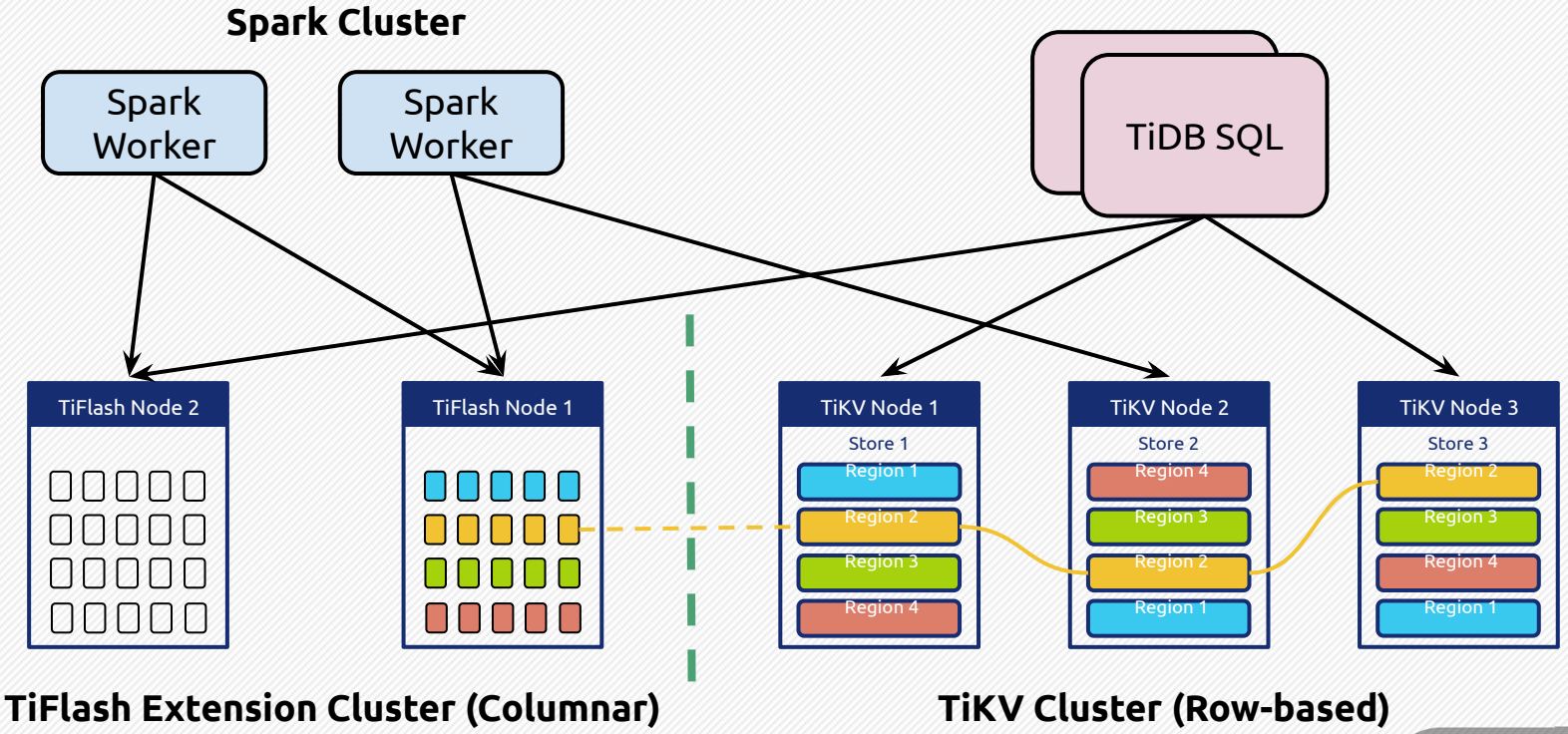
```
MySQL [tpch_001]> explain analyze select sum(l_extendedprice) / 7.0 as avg_yearly
    -> from LINEITEM, PART
    -> where
    ->     p_partkey = l_partkey and
    ->     p_brand = 'Brand#44' and
    ->     p_container = 'WRAP PKG' and
    ->     l_quantity < (
    ->         select 0.2 * avg(l_quantity)
    ->         from LINEITEM
    ->         where l_partkey = p_partkey
    ->     );
```

id	execution info
Projection_16	time:1.192774103s, loops:2, rows:1
└─StreamAgg_21	time:1.192771013s, loops:2, rows:1
└─Projection_40	time:1.192764823s, loops:2, rows:1
└─HashRightJoin_42	time:1.192739153s, loops:2, rows:1
├─HashRightJoin_26	time:582.225026ms, loops:6, rows:26
│ ├─TableReader_29	time:91.472466ms, loops:2, rows:1
│ │ └─Selection_28	time:83ms, loops:6, rows:1
│ │ └─TableScan_27	time:71ms, loops:6, rows:2000
│ └─TableReader_31	time:581.802578ms, loops:60, rows:60175
│ └─TableScan_30	proc max:539ms, min:97ms, p80:142ms, p95:539ms, rows:60175, iters:107, tasks:10
└─HashAgg_36	time:1.192447909s, loops:3, rows:2000
└─TableReader_37	time:1.190742387s, loops:19, rows:17735
└─HashAgg_32	proc max:0s, min:0s, p80:0s, p95:0s, rows:0, iters:0, tasks:10
└─TableScan_35	proc max:632ms, min:117ms, p80:171ms, p95:632ms, rows:60175, iters:60185, tasks:10

Flashback drop

```
mysql> drop table t;
Query OK, 0 rows affected (0.02 sec)

mysql> show create table t;
ERROR 1146 (42S02): Table 'test.t' doesn't exist

mysql> recover table t;
Query OK, 0 rows affected (0.12 sec)

mysql> select * from t;
...
```

TiFlash: Columnar storage for TiDB



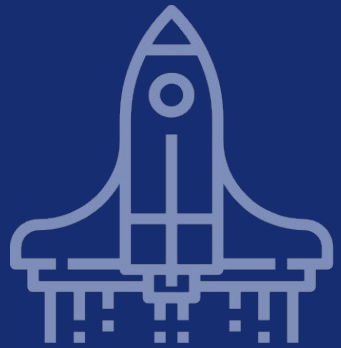
TiDB 3.0 roadmap

- Role-based Access Control
- Skyline pruning
- Cascades planner
- Table partitioning
- **Online config reloading**
- Views/window Functions
- **Plugin system**
- Vectorized execution in TiKV Coprocessor
- Query tracing
- Query plan mangement
- Index hash Join
- Radix hash Join
- **Official Jepsen Test**
- **Pessimistic transaction support**
- **Titan (a RocksDB storage plugin)**
- Fast CSV import
- ....

TiDB 3.0 GA
coming June, 2019

PingCAP

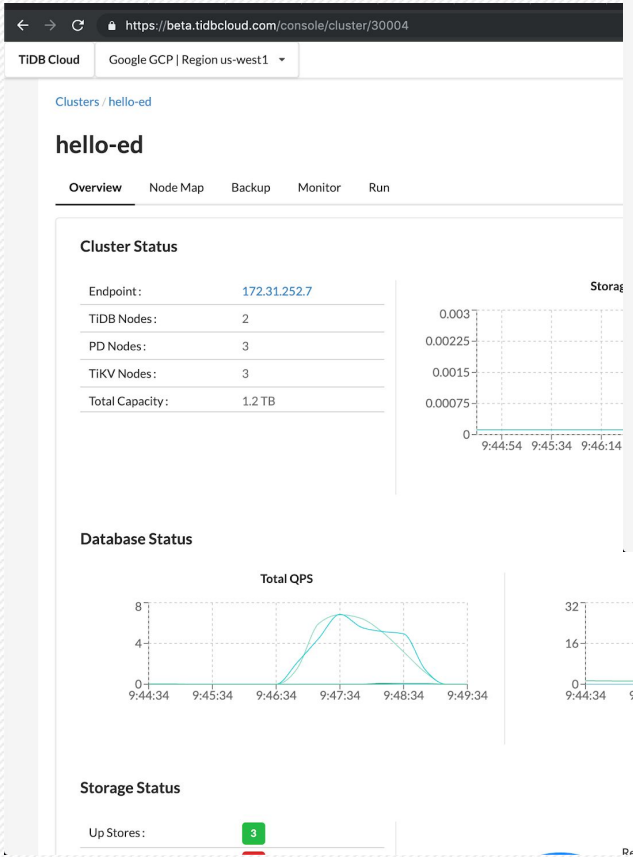
What's next?

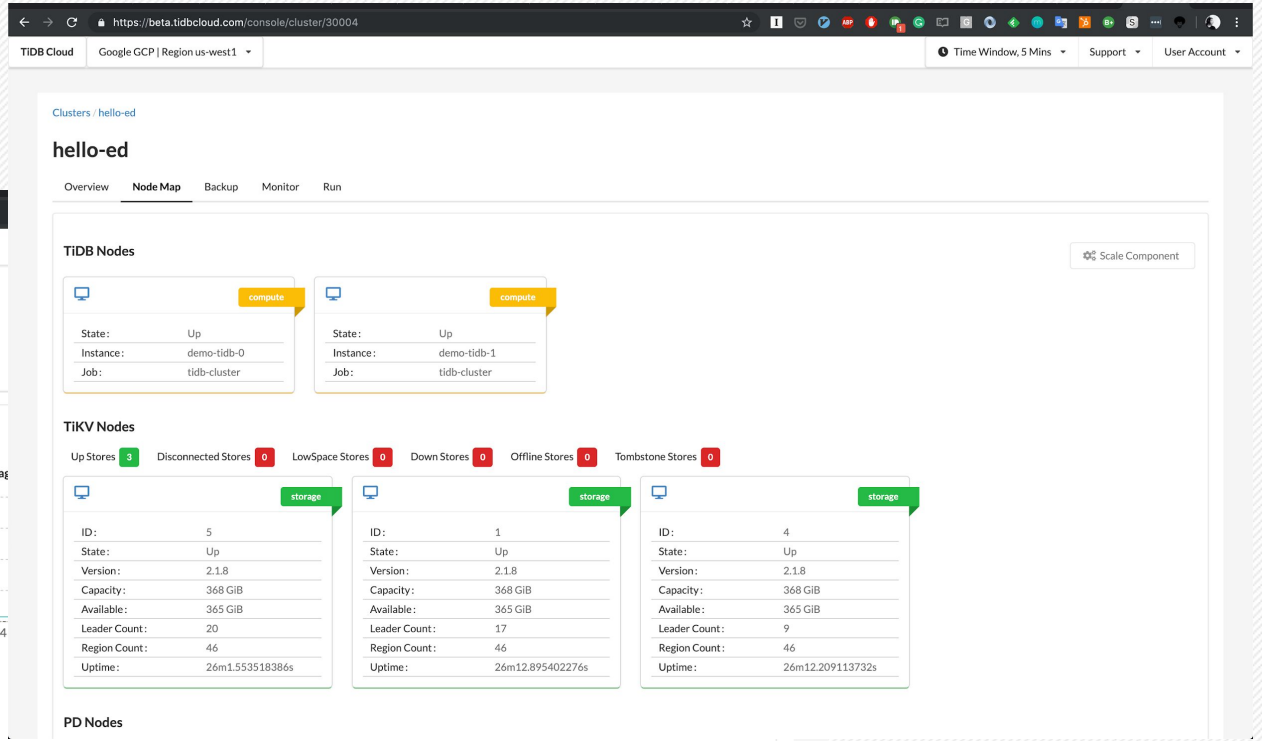


What’s next?

- Follower read (Geo-replication)
- Cascades planner
- TiDB DBaaS
- Physical backup/restore using Raft learner
- Dynamic/flexible data placement strategy
- Serverless!
- ...

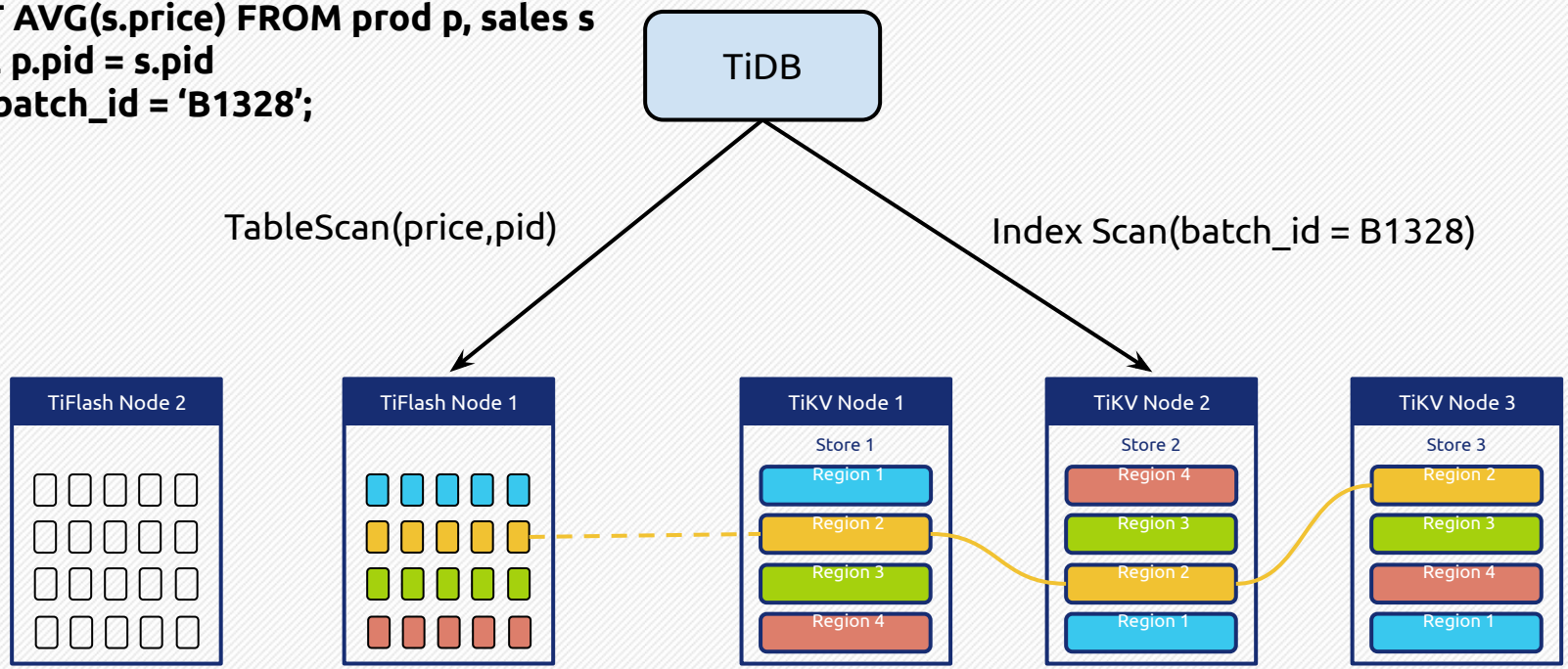
DBaaS





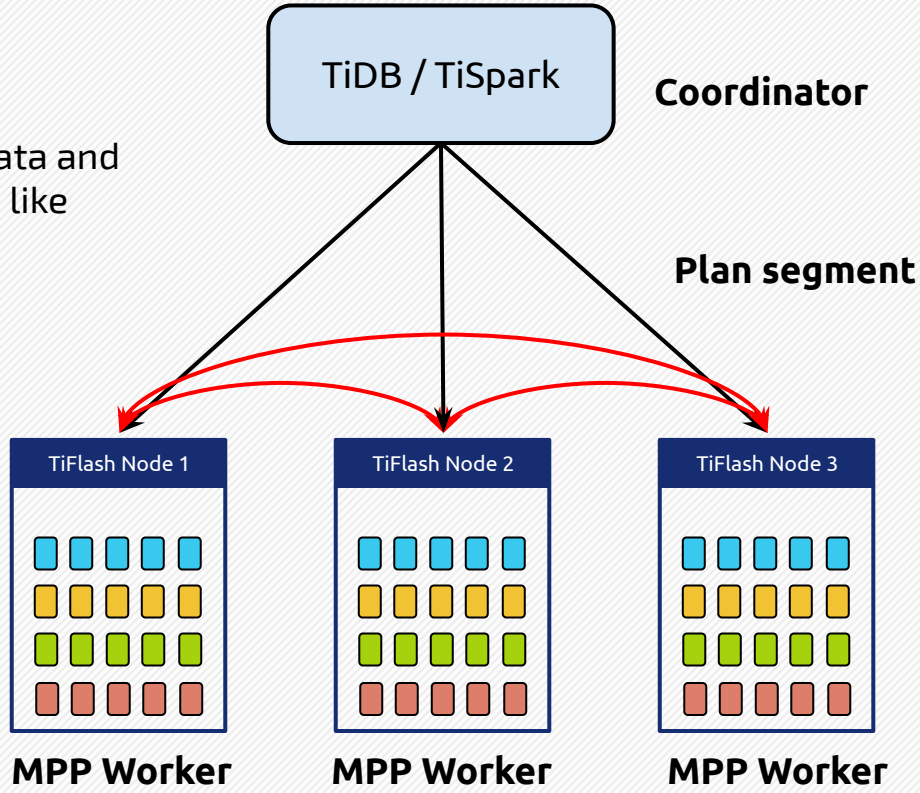
Towards real HTAP (WIP)

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



Towards real HTAP (WIP)

TiFlash nodes exchange data and enable complex operators like distributed join.



Don’t miss the TiDB track & booth (#302)😉

- TiDB and Amazon Aurora: Compare, Contrast, Combine
- Using chaos engineering to ensure system reliability
- Leveraging Intel Optane to tackle I/O challenges
- A deep look into TiDB’s SQL processing layer, optimized for a distributed system
- Introducing a new columnar storage engine (TiFlash) that makes hybrid OLTP/OLAP a reality
- Building TiDB as a managed service (aka DBaaS) on a Kubernetes Operator
- Migration best practices in and out of TiDB from MySQL and MariaDB

PingCAP

Thank you!

github.com/pingcap/tidb