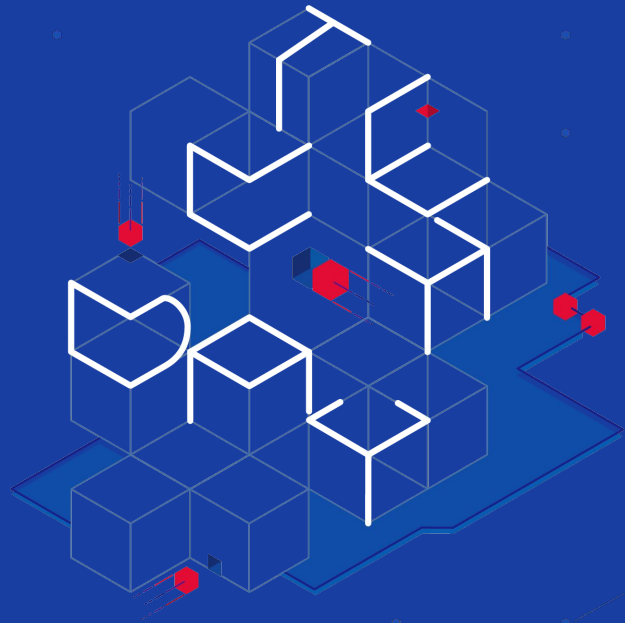
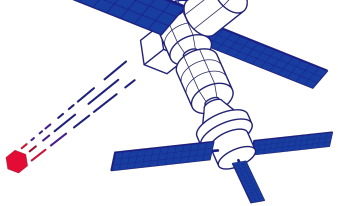
TiDB TechDay 2019

全面解析 TiDB 3.0

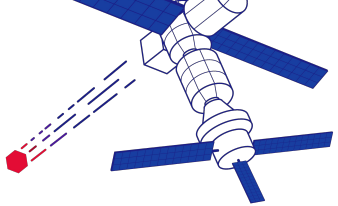
VP of Engineering
shenli@pingcap.com



Agenda

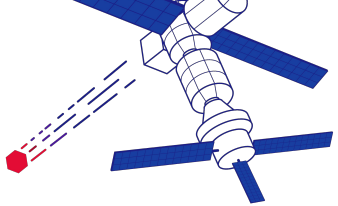


- 回顾 2.1
- 3.0 解读
  - 设计思路
  - 扩展性
  - 易用性
  - 性能
  - HTAP



从 TiDB 2.1 说起

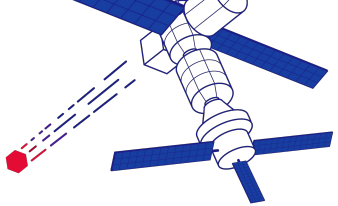
TiDB 2.1 特性



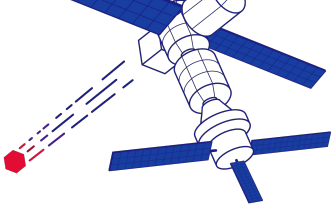
- Raft Learner
- Raft PreVote
- Hotspot scheduling
- New Aggregation Framework
- Better Optimizer
- Parallely executing DDL

Correctness, Stability, Safety

问题

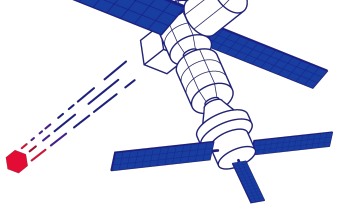


- 大集群：百 TB 数据
- 高 QPS：几十万 QPS 写入
- 突发写入：短时间产生热点
- 复杂查询：HTAP 负载

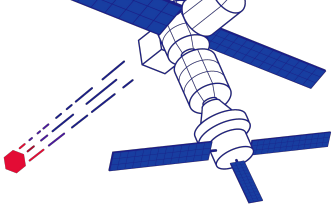


TiDB 3.0 解读

设计思路

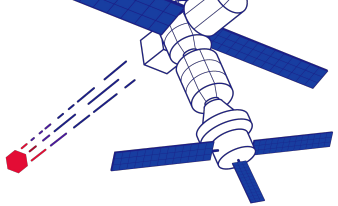


- 支持**大规模集群**（几百 TB 级别）
- 查询计划/内存使用/集群状态**稳定**
- 简单**易用**
- **性能！性能！性能！性能！**



扩展性

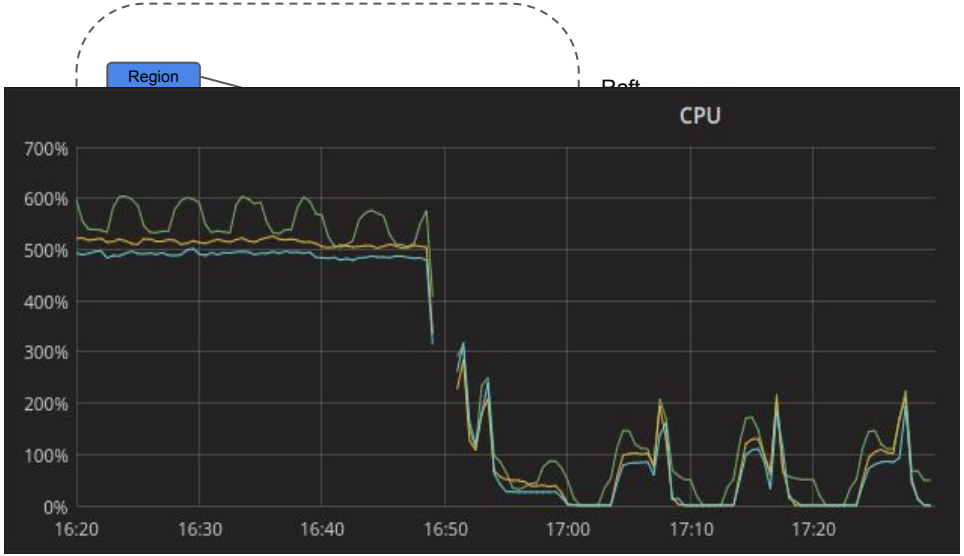
数据量大了会怎么样？



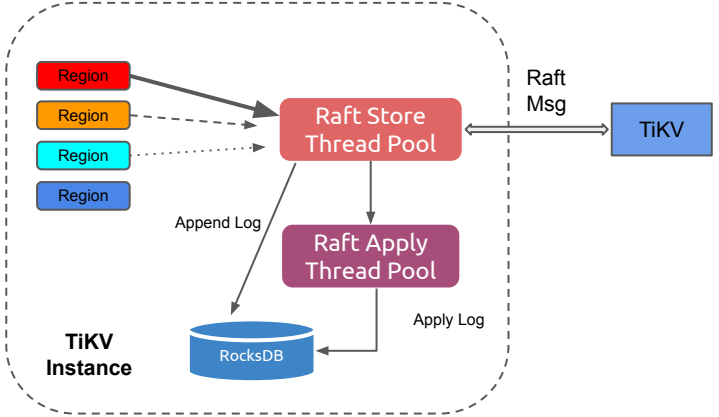
- 集群中存在大量实例+海量 Region
  - 心跳压力
  - 调度逻辑复杂
  - GC 工作量加重
- 单个实例压力大
  - Raft 模块单线程
  - 磁盘 IO 压力

Raft

- 多线程 Raft/Apply 模块



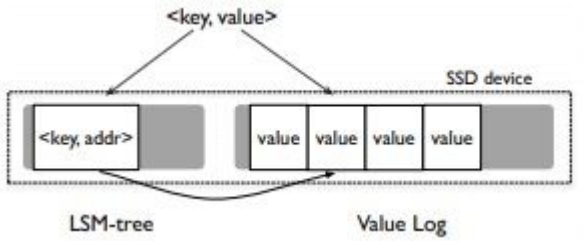
- 静默 Region

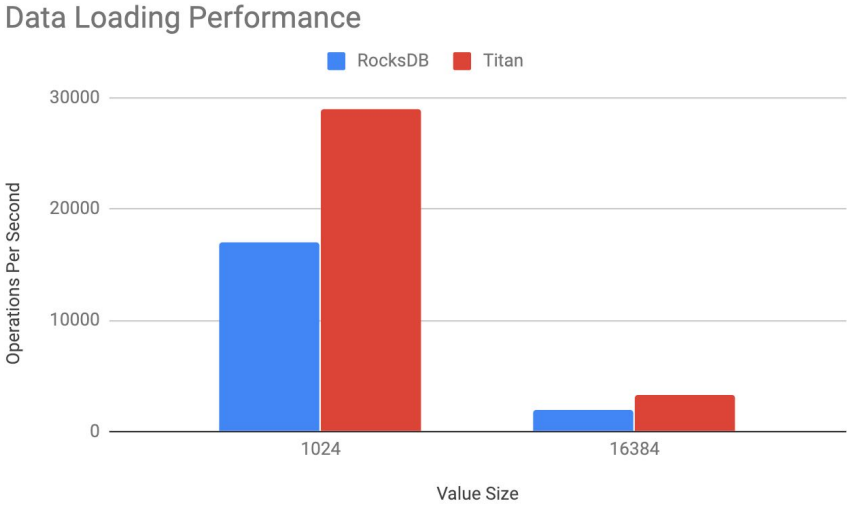


节省资源

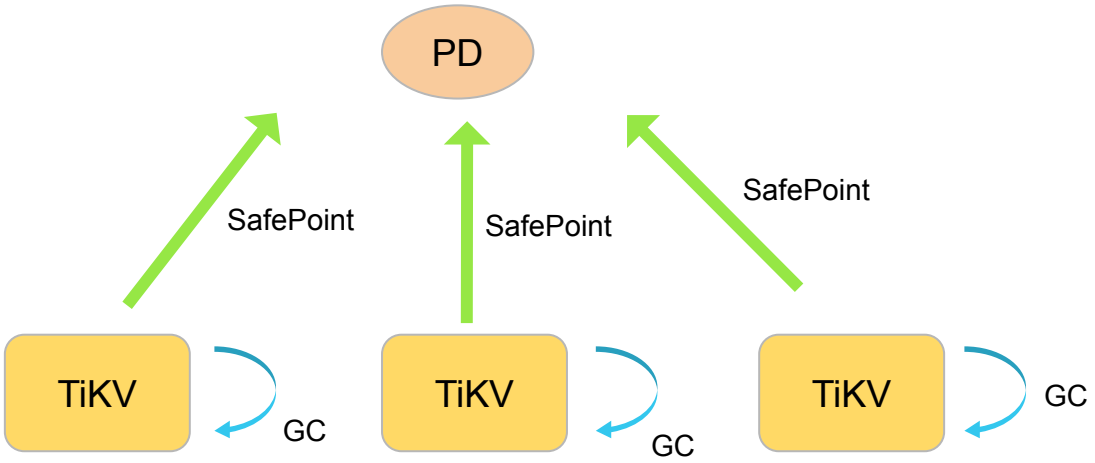
Titan

- Separating Keys from Values
  - Write amplification
  - Smaller LSM tree



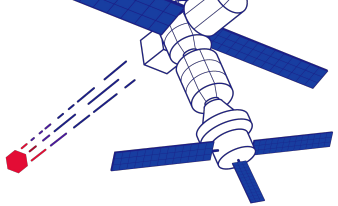


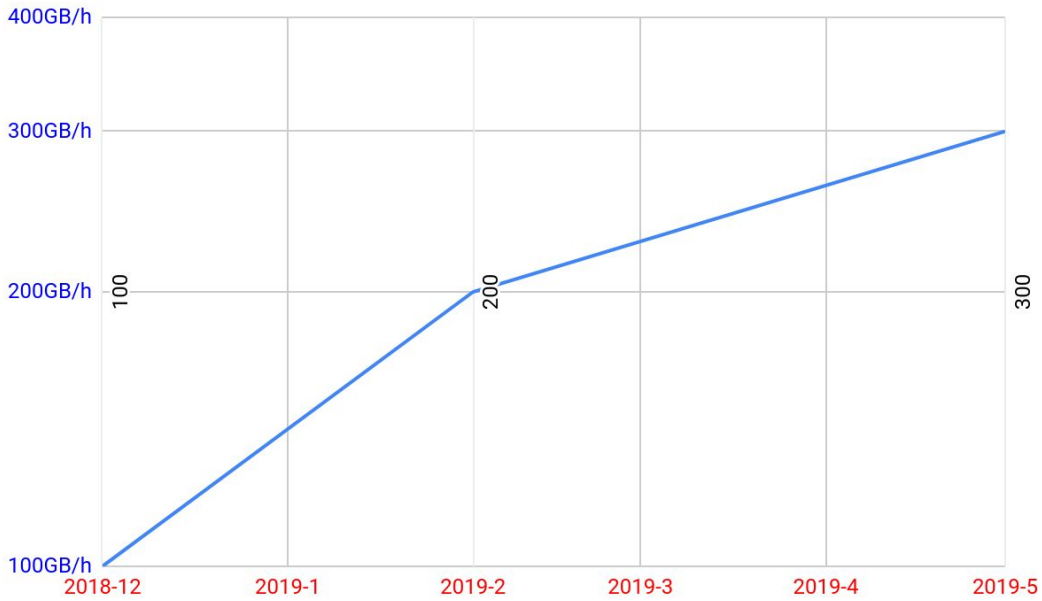
分布式垃圾回收



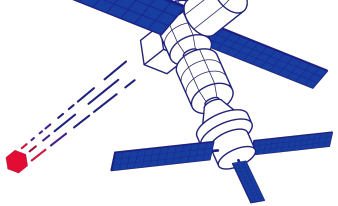
Lightning

- TiDB 专用数据加载工具
- 300 GB+/h
- 支持 CSV

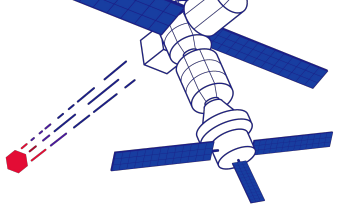




还缺什么？



- 备份和恢复效率
- 冷热数据分离
  - 成本
  - 性能
- 弹性伸缩
- 跨机房

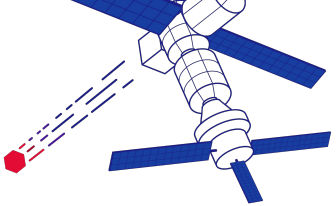


MAY

THE SCALE

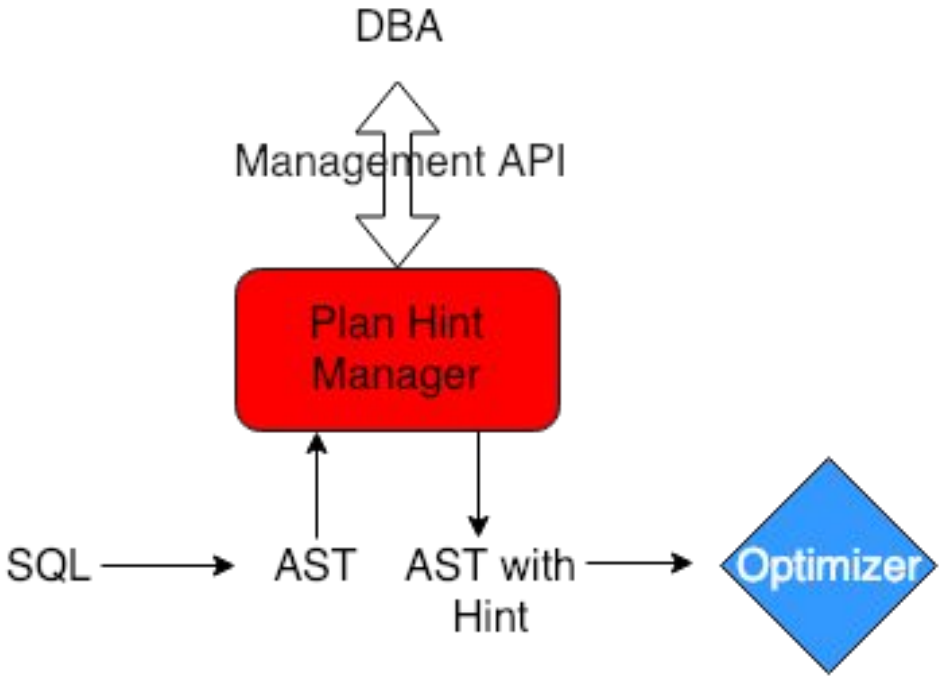
BE WITH

YOU



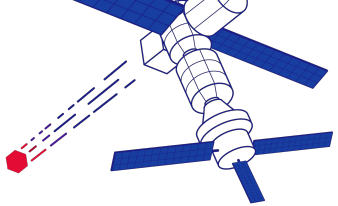
易用性

Query Plan Management



Information Schema

- Observing internal status by SQL
- Several new INFORMATION_SCHEMA tables:
  - Slow query (SLOW_QUERY)
  - Hot data range (TIDB_HOT_REGIONS)
  - Storage nodes status / Data distribution status (TIKV_STORE_STATUS)

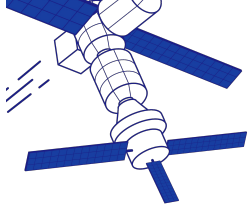


```
MySQL [INFORMATION_SCHEMA]> show tables;
+---------------------------------------+
| Tables_in_INFORMATION_SCHEMA          |
+---------------------------------------+
| ANALYZE_STATUS                        |
| CHARACTER_SETS                        |
| …                                     |
| SCHEMA_PRIVILEGES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

Example: Slow Query

Example: Show top 3 slowest queries

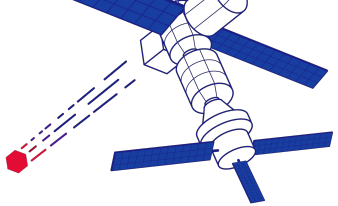
```
MySQL [(none)]> select sleep(10);
+-----------+
| sleep(10) |
+-----------+
|         0 |
+-----------+
1 row in set (10.002 sec)

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)
```

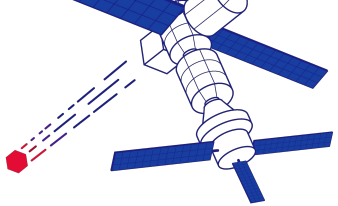
```
MySQL [INFORMATION_SCHEMA]> select Query, Query_time
    -> from INFORMATION_SCHEMA.SLOW_QUERY
    -> ORDER BY Time DESC LIMIT 3;
+-------------------+--------------+
| Query             | Query_time   |
+-------------------+--------------+
| select sleep(15); | 15.001227832 |
| select sleep(10);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

Example: Cluster Status



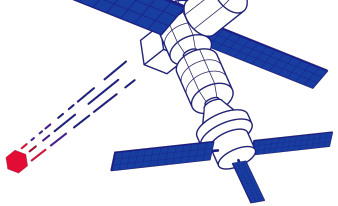
```
MySQL [INFORMATION_SCHEMA]> select STORE_ID,ADDRESS,VERSION,CAPACITY,LEADER_COUNT,REGION_SCORE,UPTIME
    -> FROM TIKV_STORE_STATUS
    -> ORDER BY STORE_ID ASC;
+----------+-----------------+-------------+----------+--------------+--------------+--------------+
| STORE_ID | ADDRESS         | VERSION     | 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



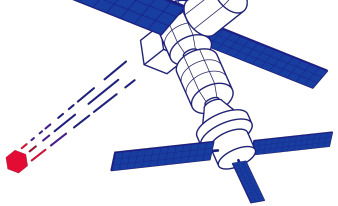
- Statistics
  - Kept more up-to-date with incremental analyze
  - Faster to generate with intelligent sampling (300 million rows: 7min 54s VS 6s)
- More optimal plan generation
  - Improvements to cost model
  - Skyline pruning
  - Join re-order
- Improved Observability
  - `EXLAIN ANALYZE` support
  - Query tracing

Explain Analyze



- EXPLAIN ANALYZE 会执行 Query，并且对每个算子输出运行时详细信息
  - **time**：该算子的运行时间
  - **loop**：被父算子调用 Next 的次数
  - **rows**：该算子返回的数据行数

Example

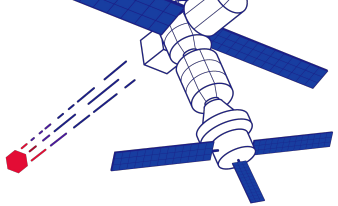


TPC-H Q17

```
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
    ->     );
```

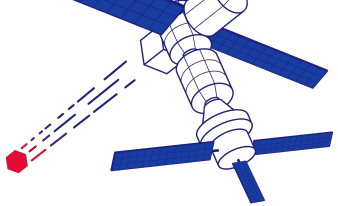
```
+-----------------------------------+-----------------------------------------------------------------------------------------------+
| id                                | execution info                                                                                |
+-----------------------------------+-----------------------------------------------------------------------------------------------+
| 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;
...
```

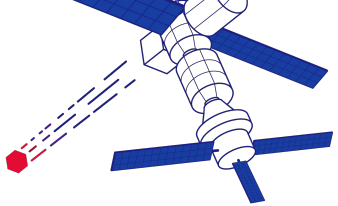
Table Partition



- Range 分区，Hash 分区
- 不支持二级分区以及分区交换
- 用处
  - 打散热点（Sequence Data）
  - 数据归档（Drop Partition）
  - 加速查询（Partition Pruning）

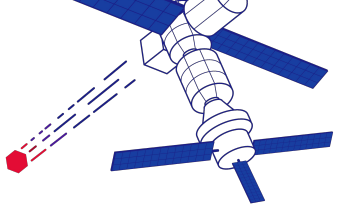
```
CREATE TABLE t (
        a INT(10) UNSIGNED NOT NULL,
        b DATETIME NOT NULL,
PRIMARY KEY (a, b)
) PARTITION BY RANGE (TO_DAYS(b))
(PARTITION p20190101 VALUES LESS THAN
(TO_DAYS('2019-01-01')),
 PARTITION p20190201 VALUES LESS THAN
(TO_DAYS('2019-02-01')),
 PARTITION p20190301 VALUES LESS THAN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

Example : Partition Pruning



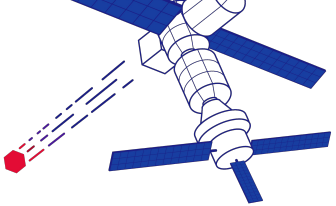
```
MySQL [test]> EXPLAIN SELECT * FROM t WHERE b < CAST('2019-04-03' AS DATETIME);
+--------------------------+---------+------+-----------------------------------------------------------------------------------------+
| id                       | count   | task | operator info                                                                           |
+--------------------------+---------+------+-----------------------------------------------------------------------------------------+
| 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)
```

为什么要提升易用性？

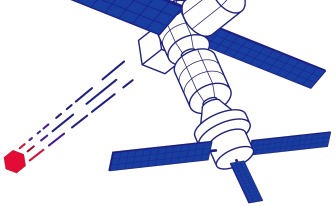


- 为开发者提供趁手的武器
  - 可以自动就避免手动
  - 能直接就不要间接
  - 有业界标准就尽量遵循
  - 以用户视角看待问题
- 开发者效率 +10%，PingCAP 效率 +1000%

下一步如何提升易用性？

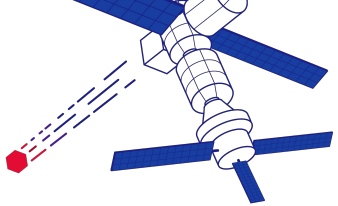


不止于代码：TiDB 社区发展及生态构建



Transaction

悲观锁



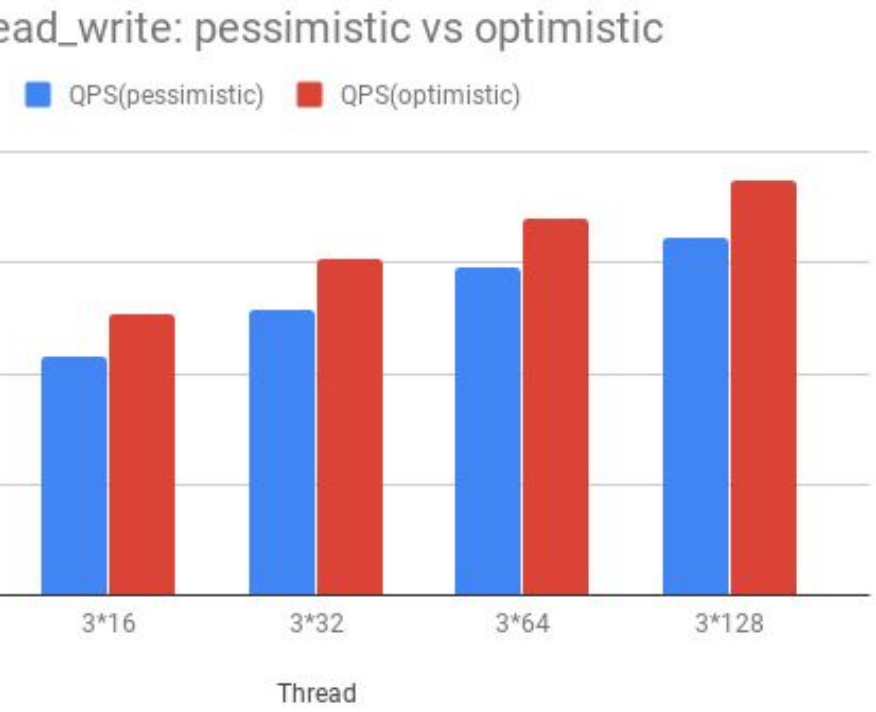
- 随 3.0.0-rc.2 作为实验性 Feature 发布
- 行为接近 MySQL
- 通过配置文件启用

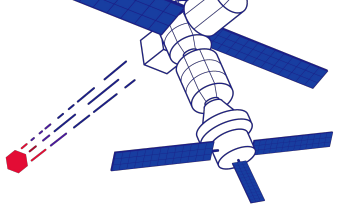
```
[pessimistic-txn]
enable = true
```

- 悲观锁/乐观锁混合使用

BEGIN /*!90000 PESSIMISTIC */; (or BEGIN PESSIMISTIC;)
SELECT …
UPDATE …
DELETE …
INSERT …
COMMIT;

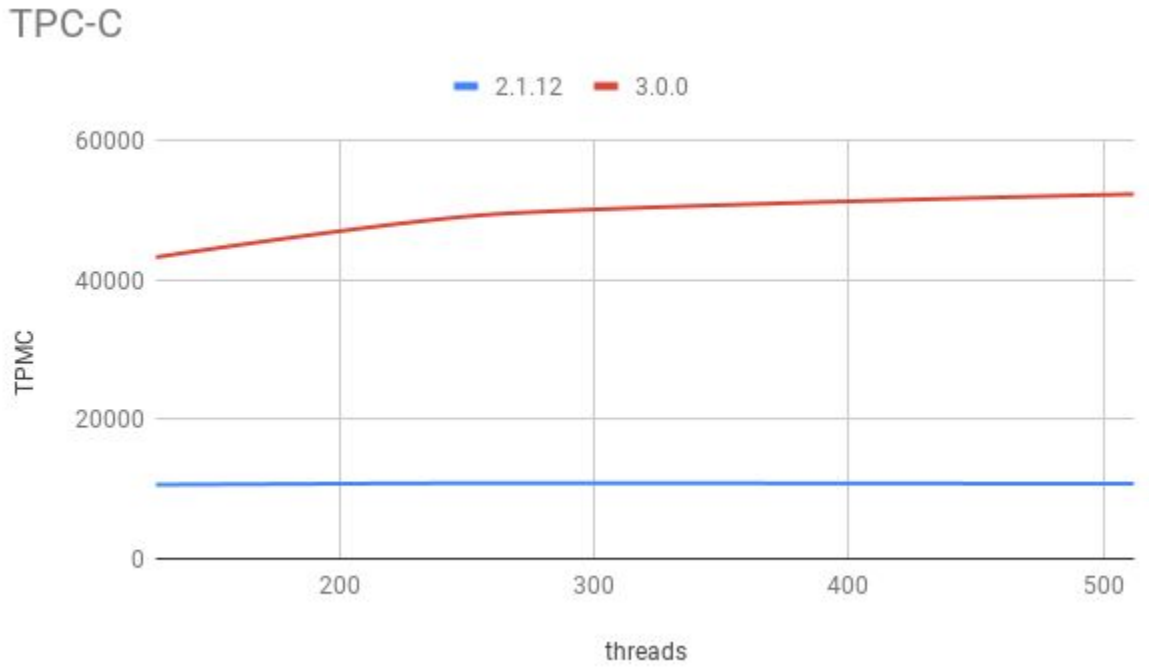
悲观锁 Benchmark



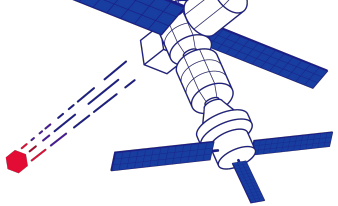


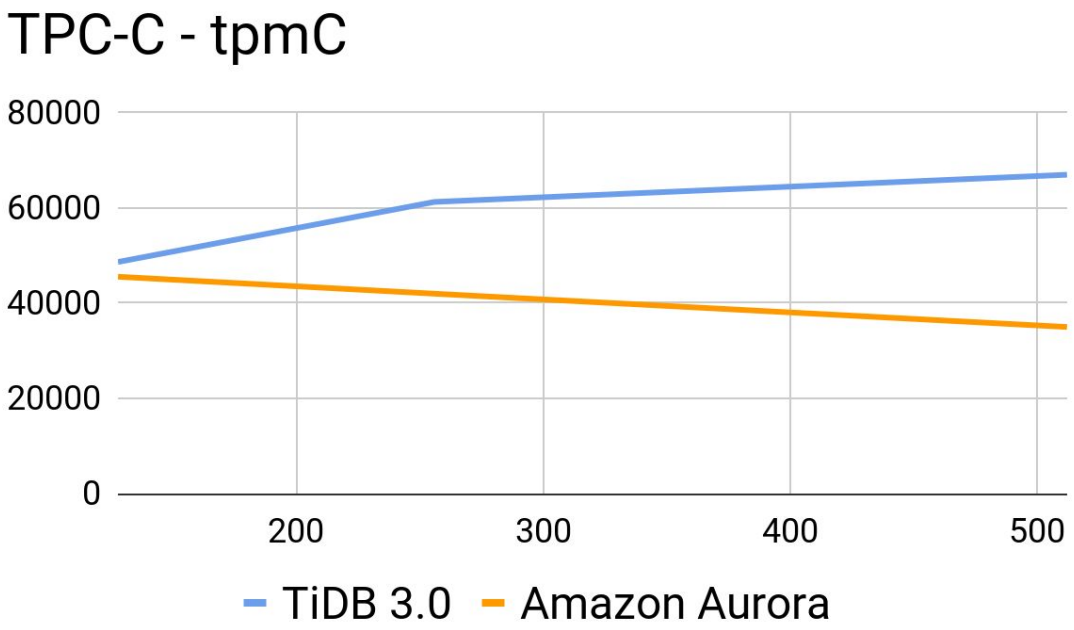
Benchmark

TPC-C: TiDB 3.0 VS 2.1



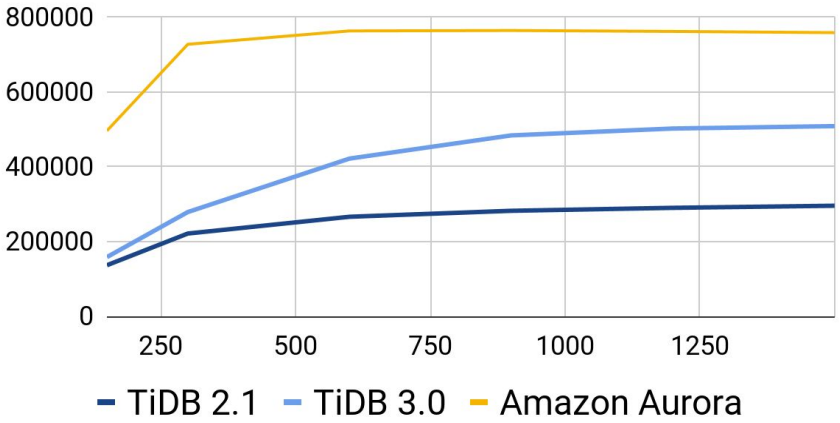
TPC-C: TiDB 3.0 VS Aurora



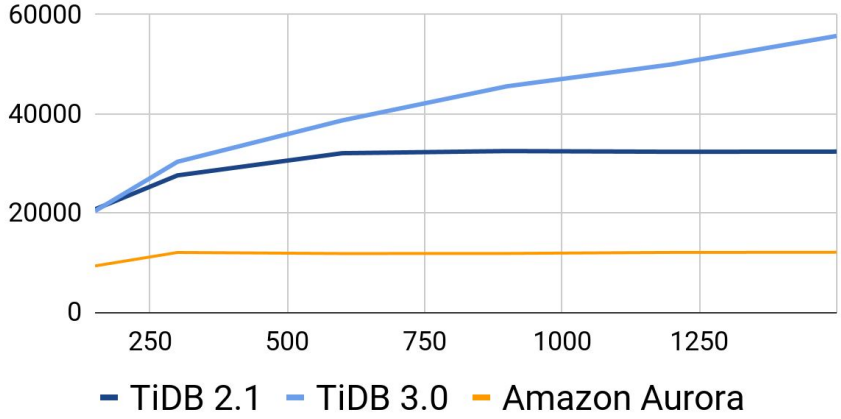


Sysbench

Sysbench - Point Select

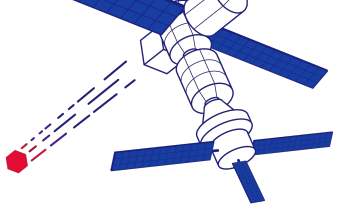


Sysbench - Update Non-Index



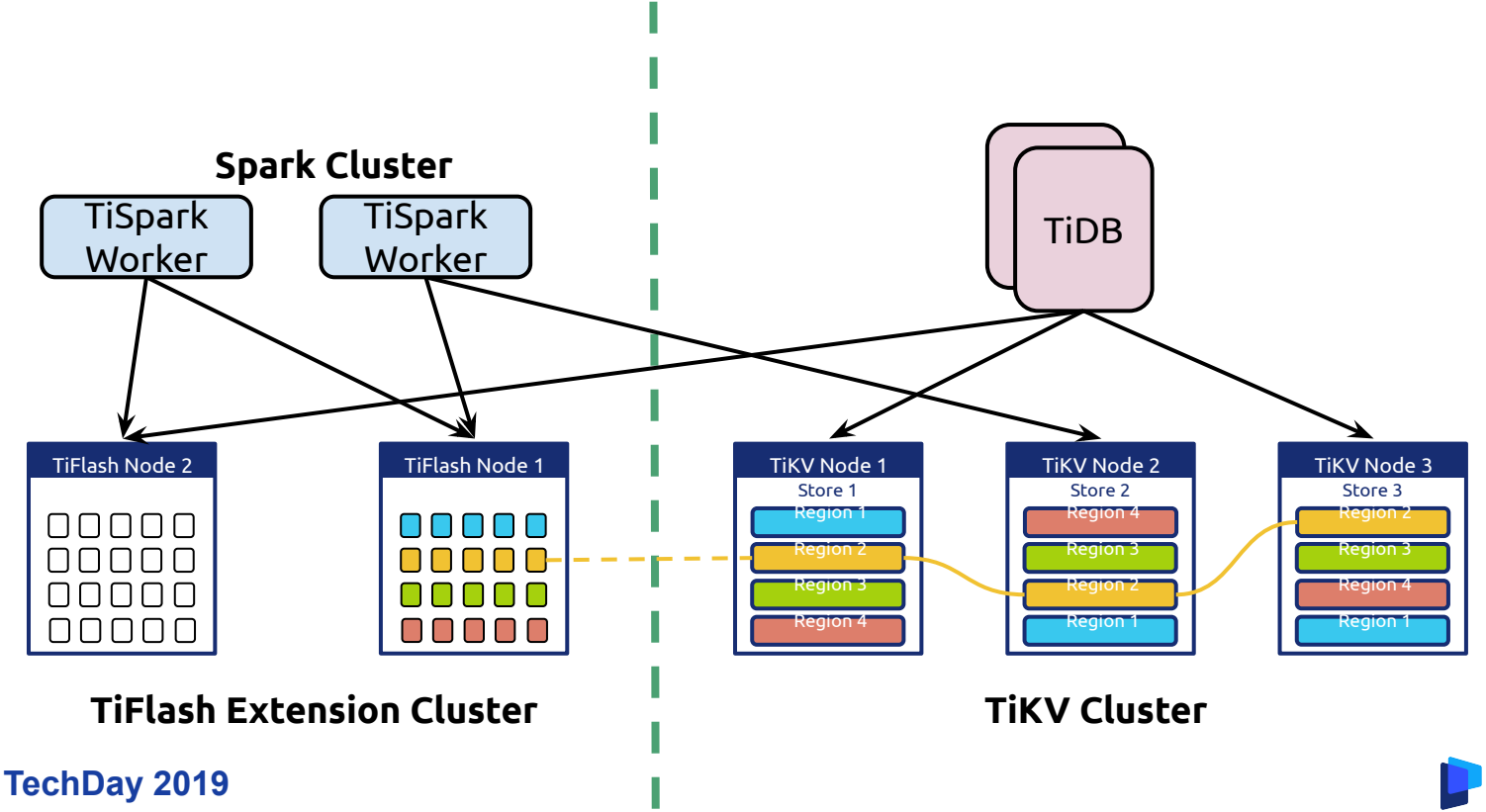
All tests are done in AWS with 3 c5d.4xlarge for tidb-server, 3 c5d.4xlarge for tikv-server.

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

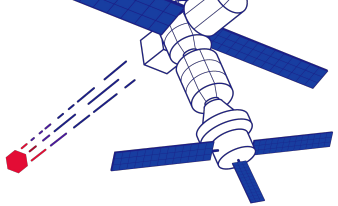


TiFlash = True HTAP

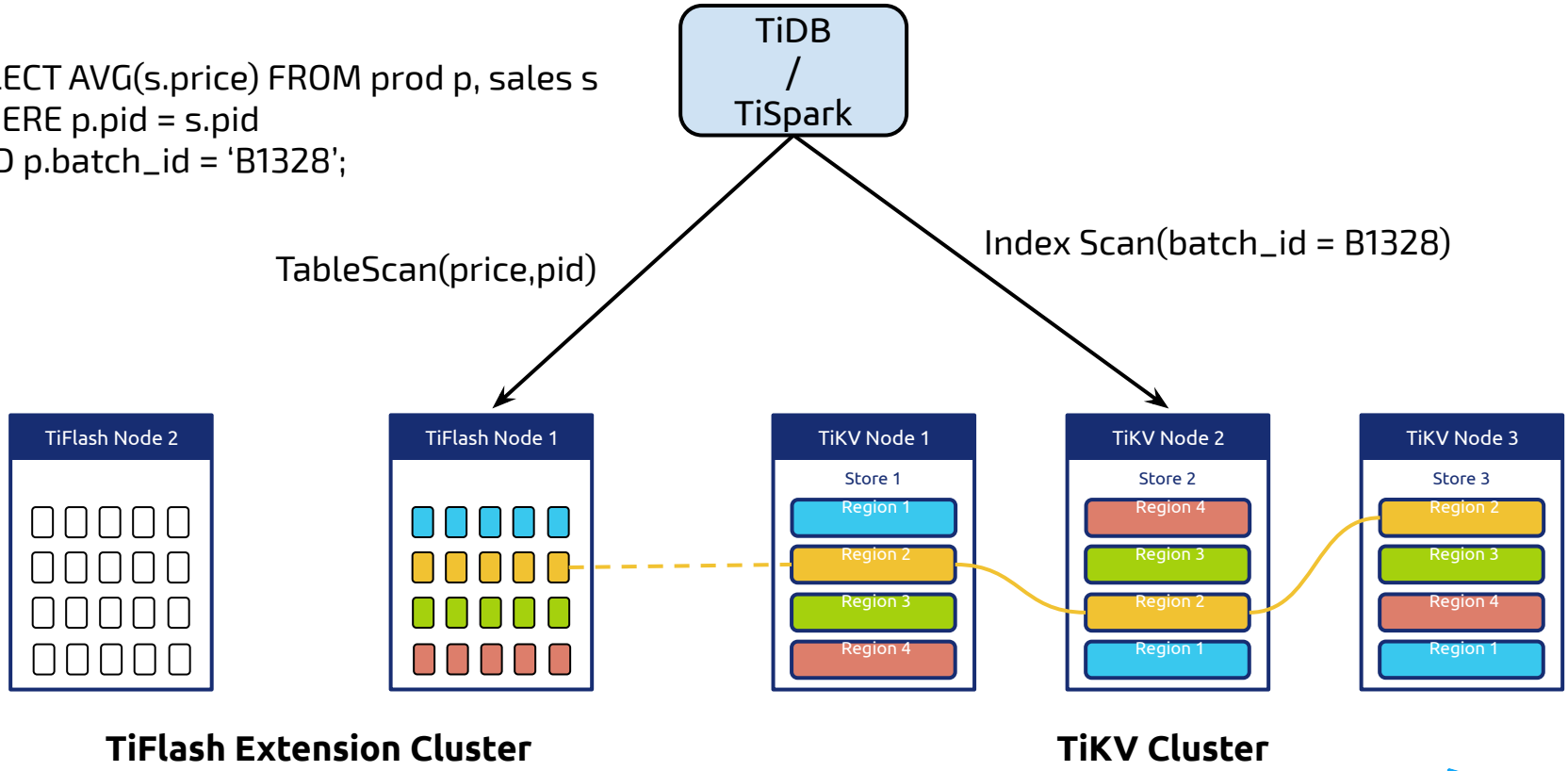
TiFlash Architecture 0.5

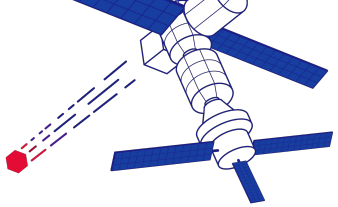


TiFlash Architecture 1.0



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





TiDB 3.0.0 于 2019 年 6 月 28 日 23:59 发布

Thank You !