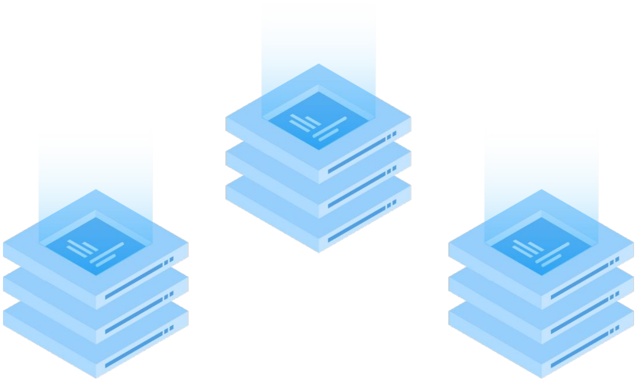
Horoscope and TiDB Query Optimizer

Jian Zhang

PingCAP TiDB



About Me

- Zhang Jian (张建), TiDB Product and Tech Manager
- zz-jason on GitHub
- Focused on:
  - Query Optimization
  - Distrubited Computation
  - Scheduling
- Email: zhangjian@pingcap.com

Part I - Background

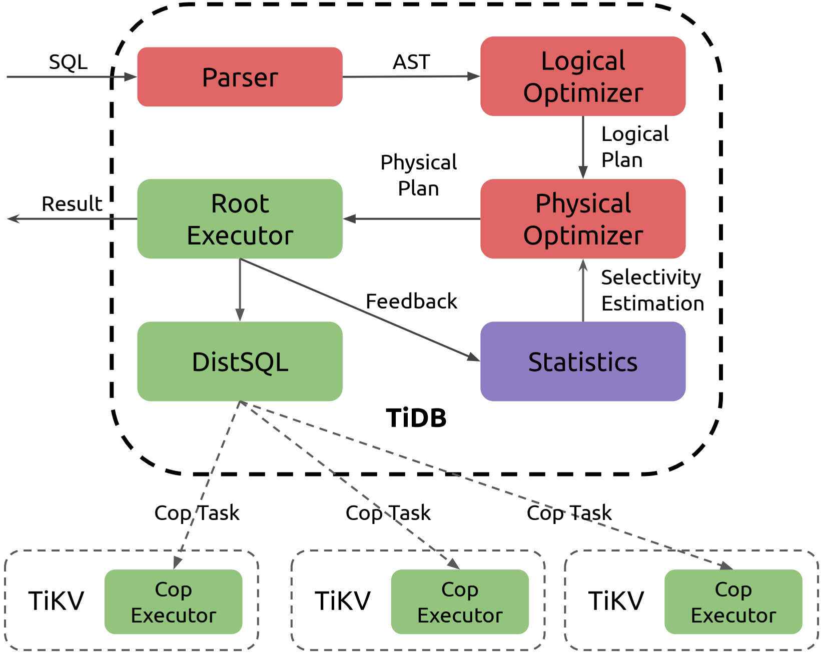
Brief Introduction to TiDB Query Optimizer

Logical Optimization: Equal, Beneficial

- Column Pruning
- Partition Pruning
- Group By Elimination
- etc.

Physical Optimization: Dynamic Programming

- Which index to choose?
- Hash Join, Merge Join, or Index Join?
- etc.



Questions About the Optimizer

How good is the query optimizer?

- What’s the percentage that the optimizer cannot choose the best index?
- Is the plan generated for a query the best one?
- How to measure the estimation errors?
- Is the optimizer better than the old version?
- etc.

Part II - Horoscope

Optimizer Test in Other DBMSs

Highly Recommend: https://github.com/zhangysh1995/awesome-database-testing

Basic Idea

Inspired by ***OptMark: A Toolkit for Benchmarking Query Optimizers*** and ***Counting, Enumerating, and Sampling of Execution Plans in a Cost-Based Query Optimizer***

Let’s enumerate all the query plans, execute them

https://github.com/chaos-mesh/horoscope

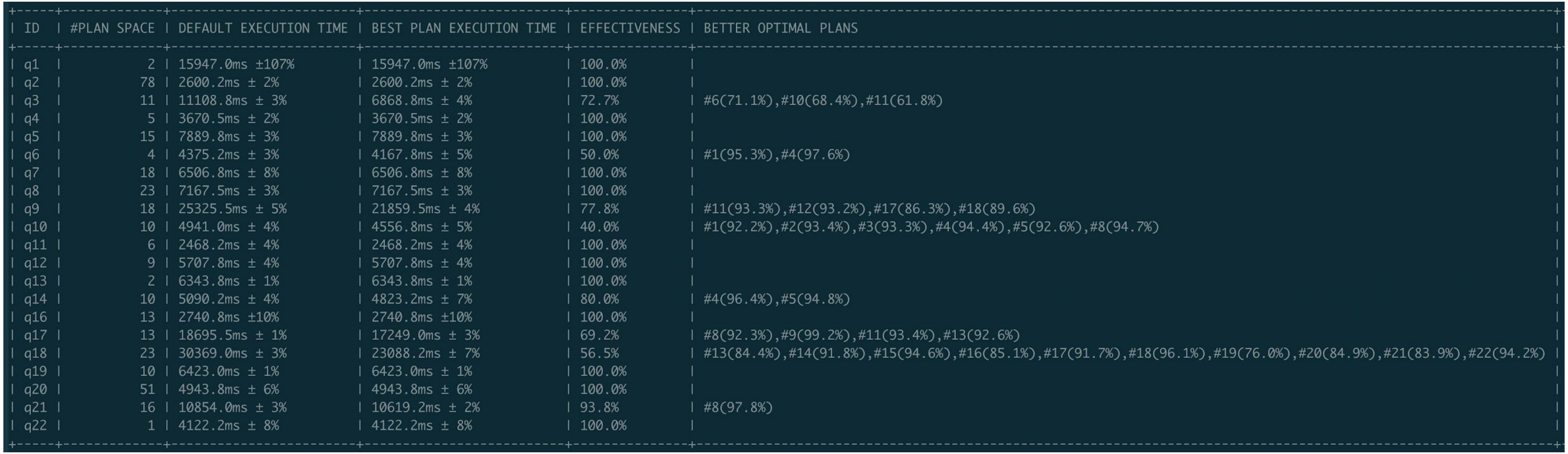
How to Enumerate All the Execution Plans

```
TiDB(root@127.0.0.1:test) > explain select /*+ nth_plan(1) */ * from t where a = 1 and b > 0 and b < 10;
+-------------------------+----------+-----------+---------------+----------------------------------------------------+
| id                      | estRows  | task      | access object | operator info                                      |
+-------------------------+----------+-----------+---------------+----------------------------------------------------+
| TableReader_7           | 0.25     | root      |               | data:Selection_6                                   |
| └─Selection_6           | 0.25     | cop[tikv] |               | eq(hehe.t.a, 1), gt(hehe.t.b, 0), lt(hehe.t.b, 10) |
|   └─TableFullScan_5     | 10000.00 | cop[tikv] | table:t       | keep order:false, stats:pseudo                     |
+-------------------------+----------+-----------+---------------+----------------------------------------------------+
3 rows in set (0.00 sec)

TiDB(root@127.0.0.1:test) > explain select /*+ nth_plan(2) */ * from t where a = 1 and b > 0 and b < 10;
+-------------------------------+---------+-----------+-------------------------+---------------------------------------------+
| id                            | estRows | task      | access object           | operator info                               |
+-------------------------------+---------+-----------+-------------------------+---------------------------------------------+
| IndexLookUp_11                | 0.25    | root      |                         |                                             |
| ├─IndexRangeScan_8(Build)     | 10.00   | cop[tikv] | table:t, index:idx_a(a) | range:[1,1], keep order:false, stats:pseudo |
| └─Selection_10(Probe)         | 0.25    | cop[tikv] |                         | gt(hehe.t.b, 0), lt(hehe.t.b, 10)           |
|   └─TableRowIDScan_9          | 10.00   | cop[tikv] | table:t                 | keep order:false, stats:pseudo              |
+-------------------------------+---------+-----------+-------------------------+---------------------------------------------+
4 rows in set (0.00 sec)
```

The First Test Report on Optimizer Effictiveness

Optimizer Effectiveness on TPC-H(SF=10) Test Report, 2020-07-08. (tidb/issues/18431)

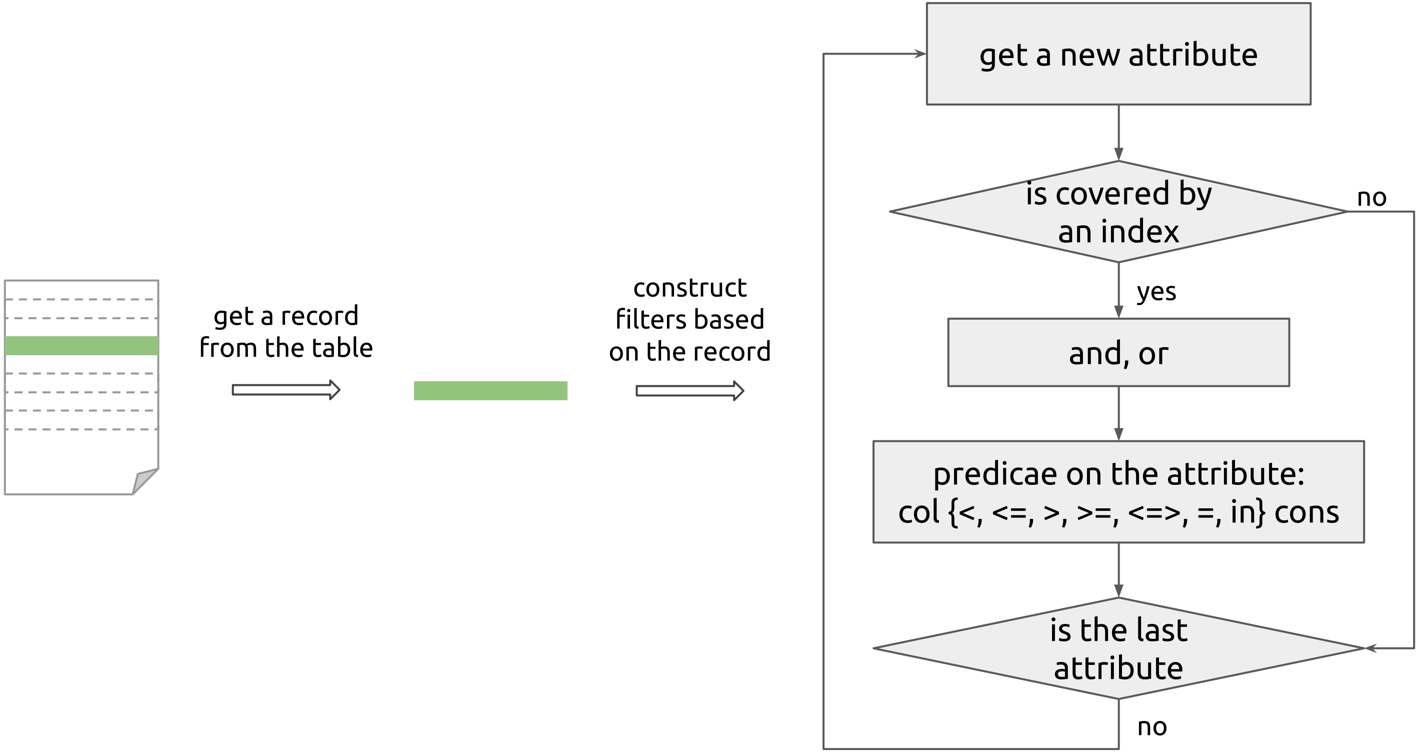


```
+-----+-------------+------------------------+--------------------------+---------------+-------------------------------------------------------------------------------------------------------------------------+
| ID  | #PLAN SPACE | DEFAULT EXECUTION TIME | BEST PLAN EXECUTION TIME | EFFECTIVENESS | BETTER OPTIMAL PLANS                                                                                                    |
+-----+-------------+------------------------+--------------------------+---------------+-------------------------------------------------------------------------------------------------------------------------+
| q1  |           2 | 15947.0ms ±107%        | 15947.0ms ±107%          | 100.0%        |                                                                                                                         |
| q2  |          78 | 2600.2ms ± 2%          | 2600.2ms ± 2%            | 100.0%        |                                                                                                                         |
| q3  |          11 | 11108.8ms ± 3%         | 6868.8ms ± 4%            | 72.7%         | #6(71.1%),#10(68.4%),#11(61.8%)                                                                                         |
| q4  |           5 | 3670.5ms ± 2%          | 3670.5ms ± 2%            | 100.0%        |                                                                                                                         |
| q5  |          15 | 7889.8ms ± 3%          | 7889.8ms ± 3%            | 100.0%        |                                                                                                                         |
| q6  |           4 | 4375.2ms ± 3%          | 4167.8ms ± 5%            | 50.0%         | #1(95.3%),#4(97.6%)                                                                                                     |
| q7  |          18 | 6506.8ms ± 8%          | 6506.8ms ± 8%            | 100.0%        |                                                                                                                         |
| q8  |          23 | 7167.5ms ± 3%          | 7167.5ms ± 3%            | 100.0%        |                                                                                                                         |
| q9  |          18 | 25325.5ms ± 5%         | 21859.5ms ± 4%           | 77.8%         | #11(93.3%),#12(93.2%),#17(86.3%),#18(89.6%)                                                                             |
| q10 |          10 | 4941.0ms ± 4%          | 4556.8ms ± 5%            | 40.0%         | #1(92.2%),#2(93.4%),#3(93.3%),#4(94.4%),#5(92.6%),#8(94.7%)                                                             |
| q11 |           6 | 2468.2ms ± 4%          | 2468.2ms ± 4%            | 100.0%        |                                                                                                                         |
| q12 |           9 | 5707.8ms ± 4%          | 5707.8ms ± 4%            | 100.0%        |                                                                                                                         |
| q13 |           2 | 6343.8ms ± 1%          | 6343.8ms ± 1%            | 100.0%        |                                                                                                                         |
| q14 |          10 | 5090.2ms ± 4%          | 4823.2ms ± 7%            | 80.0%         | #4(96.4%),#5(94.8%)                                                                                                     |
| q16 |          13 | 2740.8ms ±10%          | 2740.8ms ±10%            | 100.0%        |                                                                                                                         |
| q17 |          13 | 18695.5ms ± 1%         | 17249.0ms ± 3%           | 69.2%         | #8(92.3%),#9(99.2%),#11(93.4%),#13(92.6%)                                                                               |
| q18 |          23 | 30369.0ms ± 3%         | 23088.2ms ± 7%           | 56.5%         | #13(84.4%),#14(91.8%),#15(94.6%),#16(85.1%),#17(91.7%),#18(96.1%),#19(76.0%),#20(84.9%),#21(83.9%),#22(94.2%) |
| q19 |          10 | 6423.0ms ± 1%          | 6423.0ms ± 1%            | 100.0%        |                                                                                                                         |
| q20 |          51 | 4943.8ms ± 6%          | 4943.8ms ± 6%            | 100.0%        |                                                                                                                         |
| q21 |          16 | 10854.0ms ± 3%         | 10619.2ms ± 2%           | 93.8%         | #8(97.8%)                                                                                                               |
| q22 |           1 | 4122.2ms ± 8%          | 4122.2ms ± 8%            | 100.0%        |                                                                                                                         |
+-----+-------------+------------------------+--------------------------+---------------+-------------------------------------------------------------------------------------------------------------------------+
```

What Defines a Test Case

- Dataset: There're lots of real-world dataset on the internet.
- Schema:
  - Which type to use for an attrbute?
  - What index to construct for a table?
- Query:
  - How to generate queries based on the schema?
  - How to generate queries which cover most of the application use cases?

How to Generate Queries



How to Generate Test Cases

- Generate add-indexes DDLs

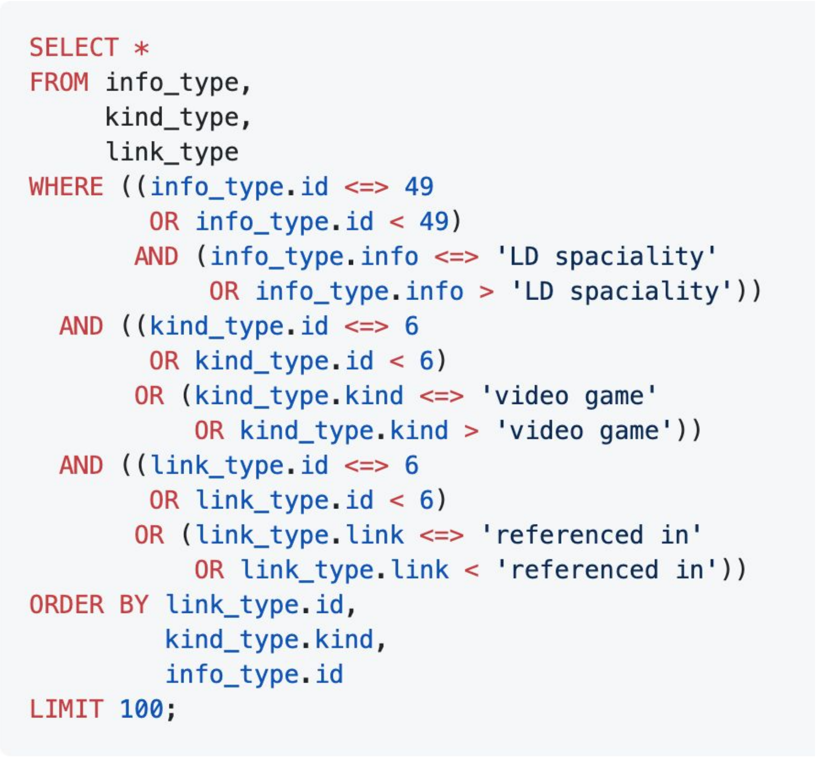
```
$ horo index gen
```

- Apply add-indexes DDLS

```
$ horo index add
```

- Generate queries

```
$ horo gen
```



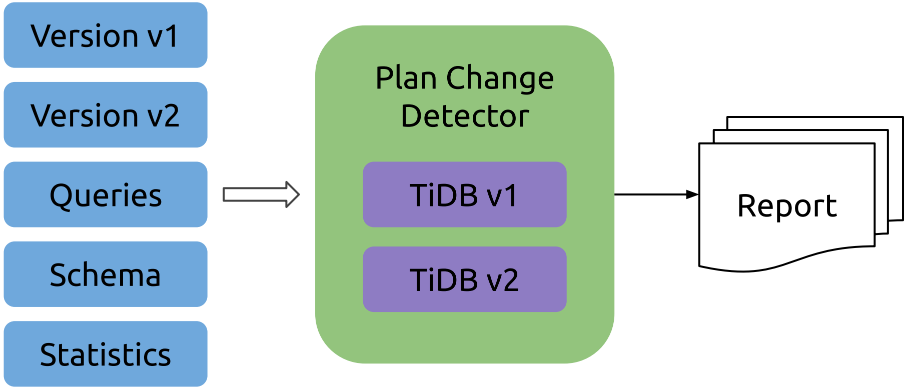
```
SELECT *
FROM info_type,
     kind_type,
     link_type
WHERE ((info_type.id <=> 49
        OR info_type.id < 49)
       AND (info_type.info <=> 'LD spaciality'
            OR info_type.info > 'LD spaciality'))
  AND ((kind_type.id <=> 6
        OR kind_type.id < 6)
       OR (kind_type.kind <=> 'video game'
           OR kind_type.kind > 'video game'))
  AND ((link_type.id <=> 6
        OR link_type.id < 6)
       OR (link_type.link <=> 'referenced in'
           OR link_type.link < 'referenced in'))
ORDER BY link_type.id,
         kind_type.kind,
         info_type.id
LIMIT 100;
```

Effectiveness when stats not up-to-date (WIP)

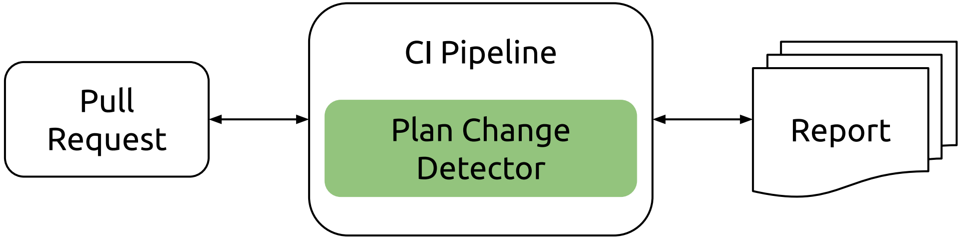
- Split database(e.g IMDB) into slices:
    - Link tables by attribute mapping (primary key ⇔ foreign key)
    - Organize tables into several groups
    - Split each group by a table or a field
- Incrementally import data slices
- Record the effectiveness metric of each round
- Measure the effectiveness changing

Plan Change Detector (WIP)

Update the cluster from version v1 to version v2:



detate plan changes in each PR:



What’s More

Reducing the selectivity estimation error: test the q-error

Learning from the sophisticated DBMS: test their effictiveness

Part III - Optimizer Improvements

Optimizer Improvements

Progress Tracking:

- Root GitHub Issue: issues/18065
- Weekly Reports: Index Selection Work Plan & Weekly Report
- Discuss Here: **#sig-planner**

TiDB Community Slack Channel
https://pingcap.com/tidbslack/

Estimation for Out-of-Bound Values (WIP)

Typically when new values are inserted after statistics collected, probably date values (issues/18461)

Old method: (ModifyRows / TotalRows) / NDV, unfriendly to small modifications

New method: 1/NDV

Avoid Independent Assumptions (WIP)

Extended Statistics (tidb/issues/18330), similar to PostgreSQL, Oracle

- `create statistics <stats_name> (<stats_type>) on <tbl> (col [, col])`
- `drop statistics <stats_name>`

Supported statistics types:

- cardinality: `where col_a > x and col_b = y`
- correlation: `where col_a > x order by col_b limit 10`

TopN, CM-Sketch and Histogram (WIP)

What are they:

- TopN: Most Common Values (MCV) in PostgreSQL
- CM-Sketch: Two-dimensional Bloomfilter with counters
- Histogram: Ordered Buckets with lower/upper bounds, and counters

Old method:

- TopN is calculated from Histogram (issues/17467), not removed from Histogram

New method:

- Extract TopN from the data scanned
- Construct CM-SKetch/Histogram without TopN values

Part IV - Join Us!

SQL Engine Team

Improve Optimizer Effictiveness

- Statistics Refactoring
- Index Selection
- Join Order Improvement
- Fast Analyze
- Query Feedback
- Plan Cache
- SQL Plan Management
- Index Tuning Advisor

Email: zhangjian@pingcap.com

Build a Fast Query Execution Engine

- Memory Management
- User-Defined Function
- OLTP/OLAP Performance Optimization
- DML Performance Optimization
- Read-Only or Read-Mostly Table Optimization
- Support More Expressions/Executors on Coprocessor

Quality & Efficiency

Build a Reliabal Database

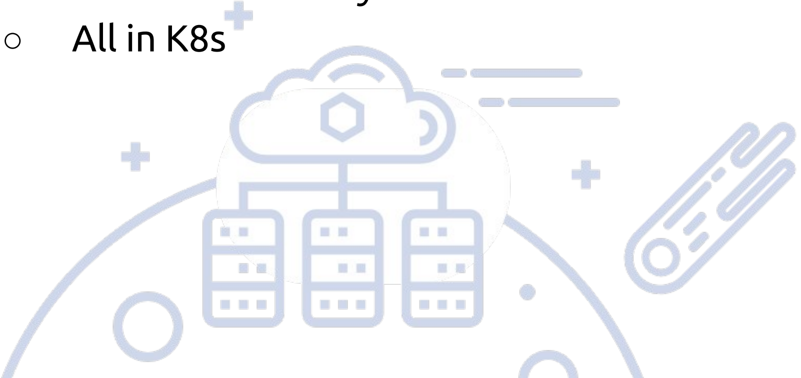
- Dive into TiDB Inplementation, and Try to Destroy
  - TiDB
  - TiFlash
  - Tools
  - DBaas
- New Test Method like Horoscope

Improve Test Efficiency

- Build Effective Test Tools
  - Chaos Mesh®
  - Failpoint
  - Jepsen
- Build an Effective Test Framework, From CI/CD to End to End Test
  - Automation Everythind
  - All in K8s

Email: zhangjian@pingcap.com

PingCAP TiDB



PingCAP

Thank you!

直播结束后我们会在交流群中分享本期 PPT

还没有加交流群的小伙伴，

可以扫码右边二维码进群👉

发现更多干货文章 & 视频：

微信公众号：PingCAP

知乎专栏：TiDB 的后花园

Bilibili：TiDB_Robot