



开源分布式数据库 RadonDB

技术揭秘

演讲人：李志昂

目录

- ▶ RadonDB架构
- ▶ Radon介绍
- ▶ Xenon介绍
- ▶ 其他功能介绍



RadonDB

- ▶ Radon: 氡, 惰性气体
- ▶ 官网:<http://radondb.io>
- ▶ 开源:<https://github.com/radondb>
- ▶ RadonDB = radon + xenon

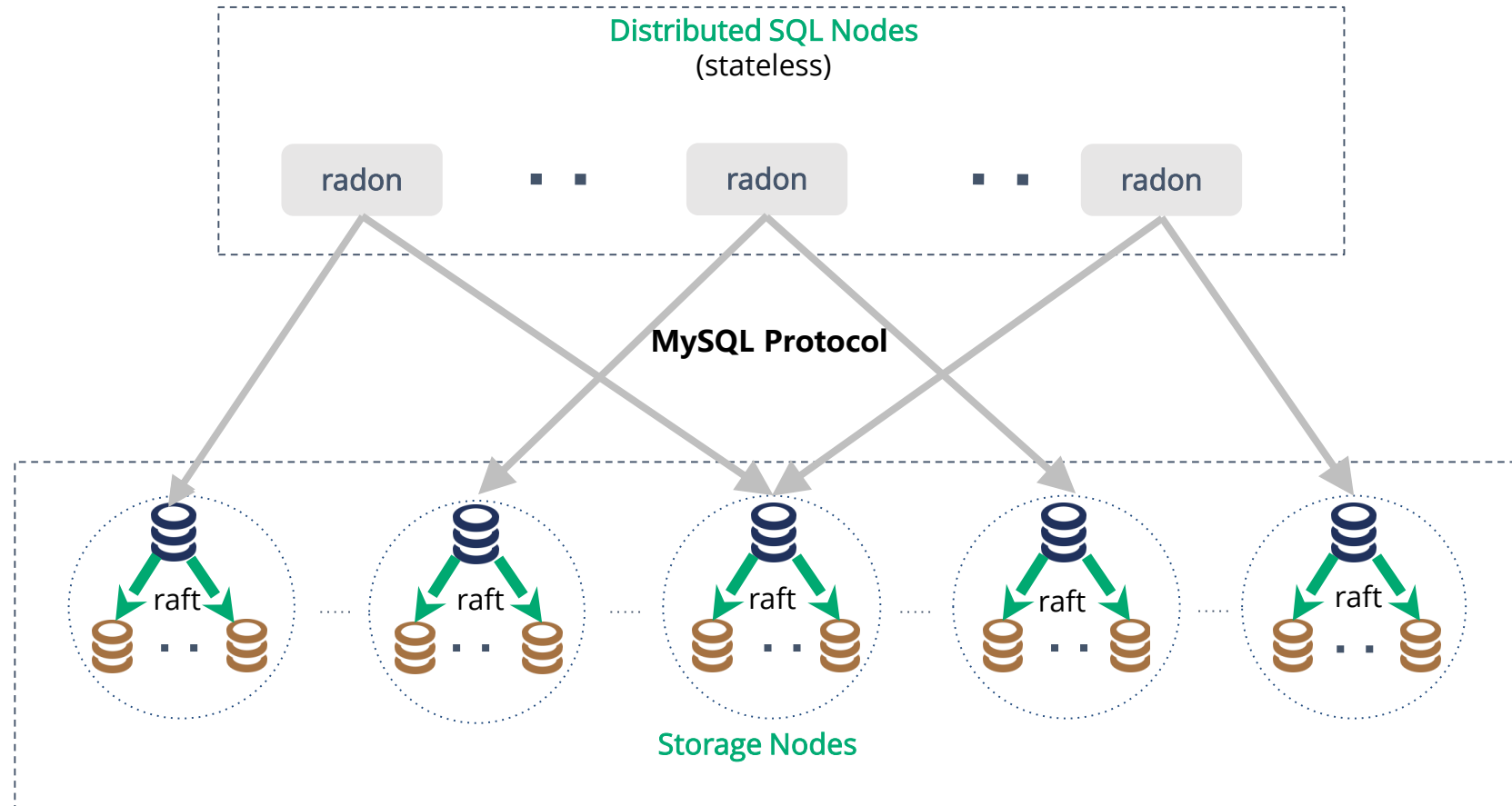


QINGCLOUD



RadonDB

Architecture



存储和计算节点的选择

- ▶ 为什么不是KV? **MySQL!**
- ▶ 稳定可靠、多索引写原子保证
- ▶ 计算下推, 数据就近计算原则
- ▶ 不仅存储还有计算能力
- ▶ SQL 与 Storage 数据传输最小化
- ▶ MySQL 8.0更加强大...

VOLTD

Why VoltDB? Product Company Customers

| SQL VS NOSQL

FoundationDB's Lesson: A Fast Key-Value Store is Not Enough

The sale and subsequent closure of FoundationDB cut short something of a grand experiment. FoundationDB, conceived as a Key-Value store, had decided to add flexibility in the form of programming and query-model "Layers" on top of its core KV store. First up was SQL, software that ran on top of core FoundationDB and provided SQL relations, indexes and queries. A graph interface and possibly other "Layers" would follow.

So how did the SQL system work out? **Running sysbench, FDB-SQL was less than half as fast as MySQL on a single machine.** This is by their own measurements. They do claim their system scales well as you add more machines, but crucially, they don't give actual numbers for the distributed SQL performance on sysbench, just "normalized performance." I'd bet some money that absolute performance was not good. If the network overhead of the loopback interface was bottlenecking the single-machine test, then an actual network with actual latency couldn't have helped.

<https://www.voltdb.com/blog/2015/04/01/foundationdb-lesson-fast-key-value-store-not-enough/>

Spanner: Becoming a SQL System

David F. Bacon Nathan Bales Nico Bruno Brian F. Cooper Adam Dickinson
Andrew Fikes Campbell Fraser Andrey Gubarev Milind Joshi Eugene Kogan
Alexander Lloyd Sergey Melnik Rajesh Rao David Shue Christopher Taylor
Marcel van der Holst Dale Woodford
Google, Inc.

ABSTRACT

Spanner is a globally-distributed data management system that

this paper, we focus on the "database system" aspects of Spanner, in particular how query execution has evolved and forced the rest of Spanner to evolve. Most of these changes have occurred since

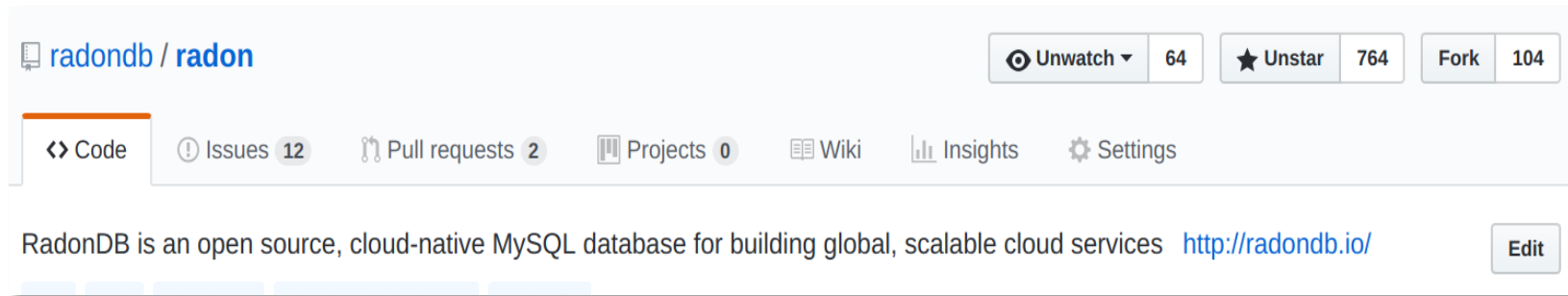


QINGCLOUD



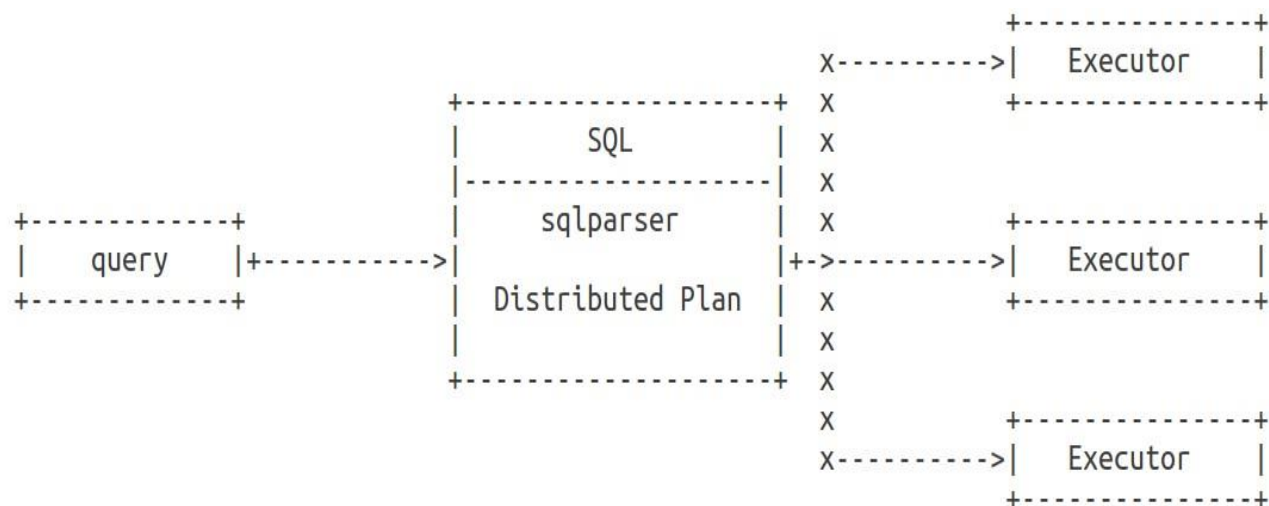
RadonDB

Radon



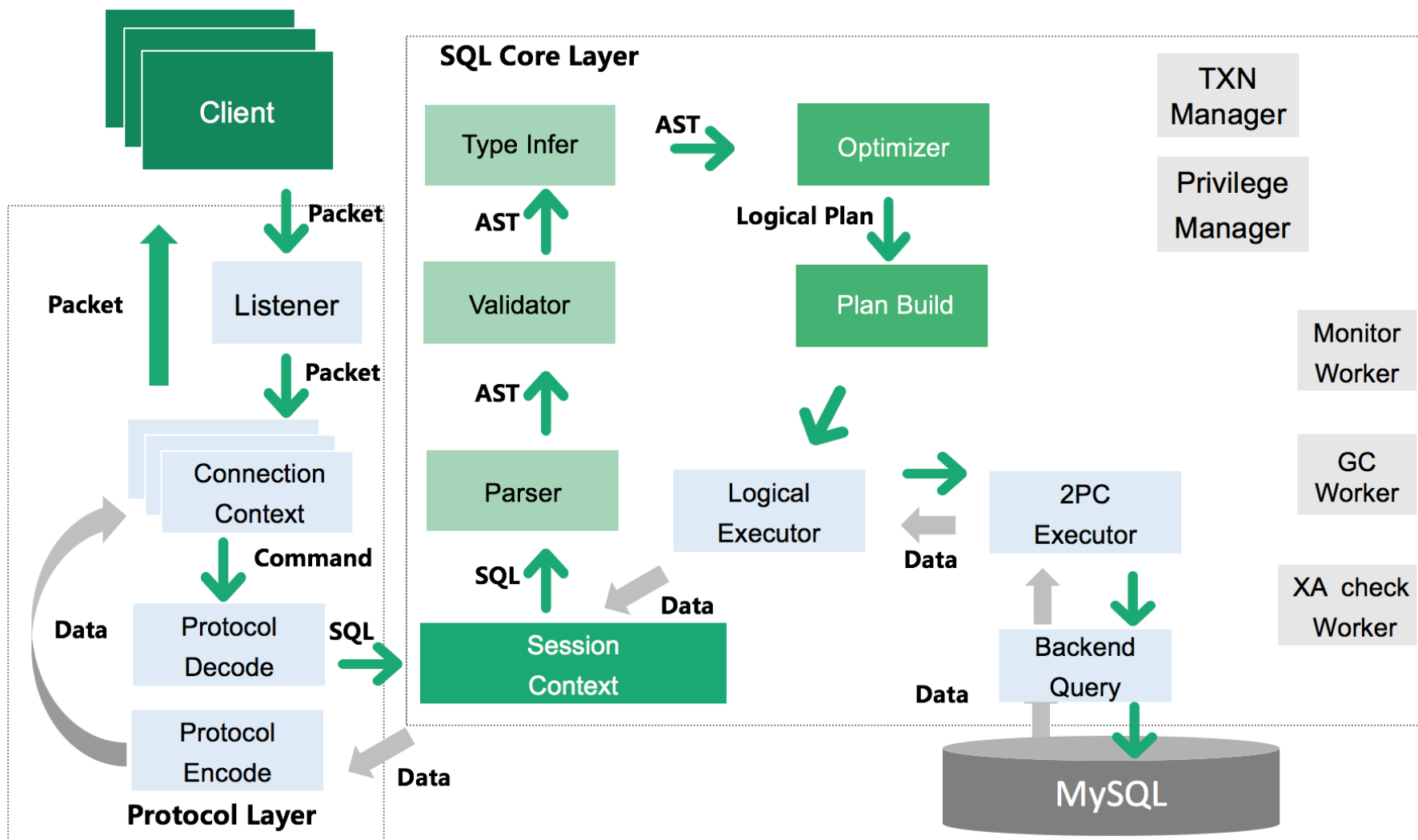
Distributed SQL

- ▶ 生成分布式执行计划
- ▶ 执行器并行执行
- ▶ orderby/limit/groupby/aggr ...
- ▶ 主从模式



分布式SQL剖析

- ▶ 语法解析
- ▶ 执行计划
- ▶ 并发执行
- ▶ 事务封装

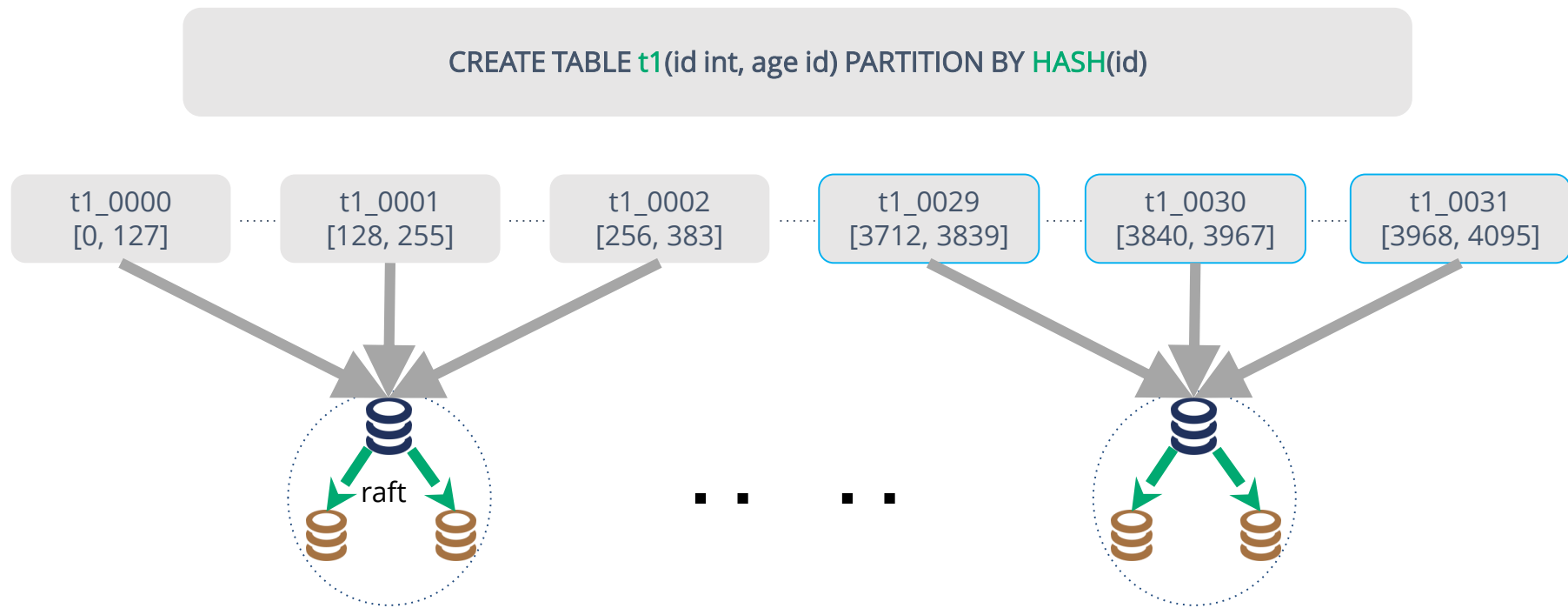


QINGCLOUD



RadonDB

数据分布



► 整张表共 4096 slots

► 每个小表 128 slots

► 小表均匀分散在 node 节点



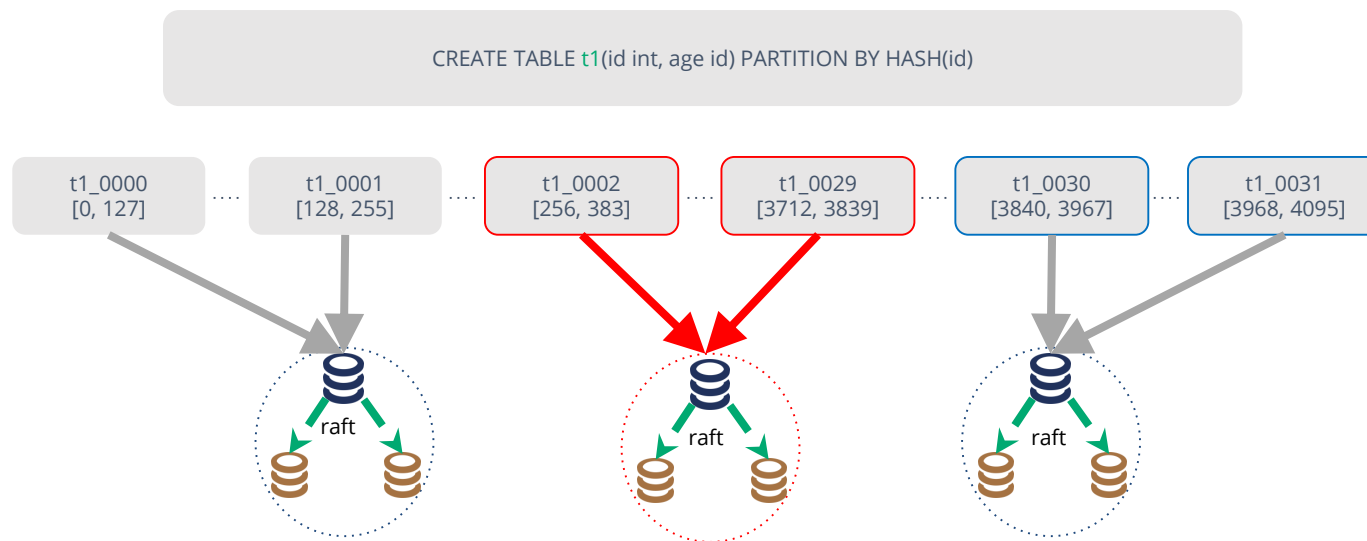
QINGCLOUD



RadonDB

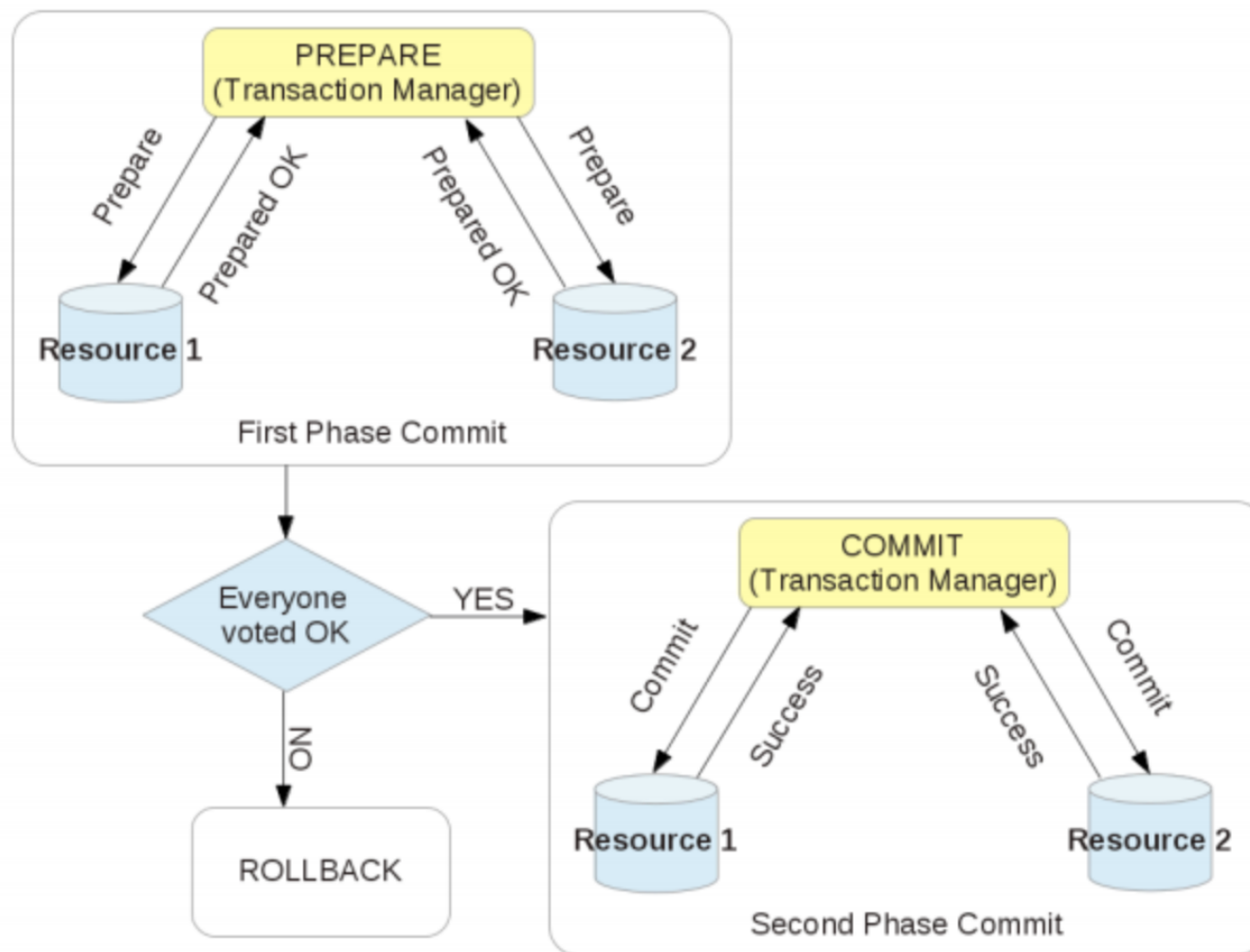
扩容

- ▶ 小表可动态漂移
- ▶ 先全量后增量
- ▶ 较大/热度高者优先
- ▶ 资源分配最优化



Distributed Transaction

- ▶ 两阶段提交算法
- ▶ 事务管理器: Radon
- ▶ 资源管理器: MySQL
- ▶ 分布式事务对用户透明



针对各种异常处理

- ▶ prepare失败/超时
 - ▶ TM发送rollback到所有RM
- ▶ 提交事务之前，某个RM返回操作错误
 - ▶ TM发送rollback到所有RM
- ▶ TM crash
 - ▶ 多种阶段不同处理，报警人工干预
- ▶ RM crash
 - ▶ 有记录xa log，最终生效
- ▶ 日志巡检等措施



分布式事务隔离级别

- ▶ 默认快照SI隔离级别
 - ▶ 所有修改只在提交时才对外可见
 - ▶ 未提交不可见
 - ▶ 部分提交不可见
- ▶ 最高可达到serializable

```
client1> select * from t1 where id>0;  
client2> update t1 set a=1 where id>0;
```

```
client1 got 1:  
case1. time -----+----->  
           |->client2-update |->client1-select
```

```
client1 got 0:  
case1. time -----+----->  
           |->client1-select |->client2-update  
  
case2. time ----->  
           |->client1-select  
           |->client2-update  
  
case3. time ----->  
           |->client1-select  
           |->client2-update
```



隔离级别验证

- ▶ xelabs/go-jepsen
- ▶ 1个更新线程，16个扫表线程
- ▶ 100多亿次操作和检测
- ▶ 各种异常情况的模拟验证

```
Thread1:  
update jepsen_si set score=0;
```

```
ThreadN:  
select score from jepsen_si;  
for cur := row.next() {  
    if pre != cur {  
        errors++  
    }  
}
```

```
time      thds  
[3599s] [r:16,u:1]
```

```
w-ops  
80000
```

```
r-ops  
3310000
```

```
error(s)  
0
```

```
total-ops  
11799980000
```

```
time      thds  
[3600s] [r:16,u:1]
```

```
w-ops  
70000
```

```
r-ops  
3130000
```

```
error(s)  
0
```

```
total-ops  
11803180000
```



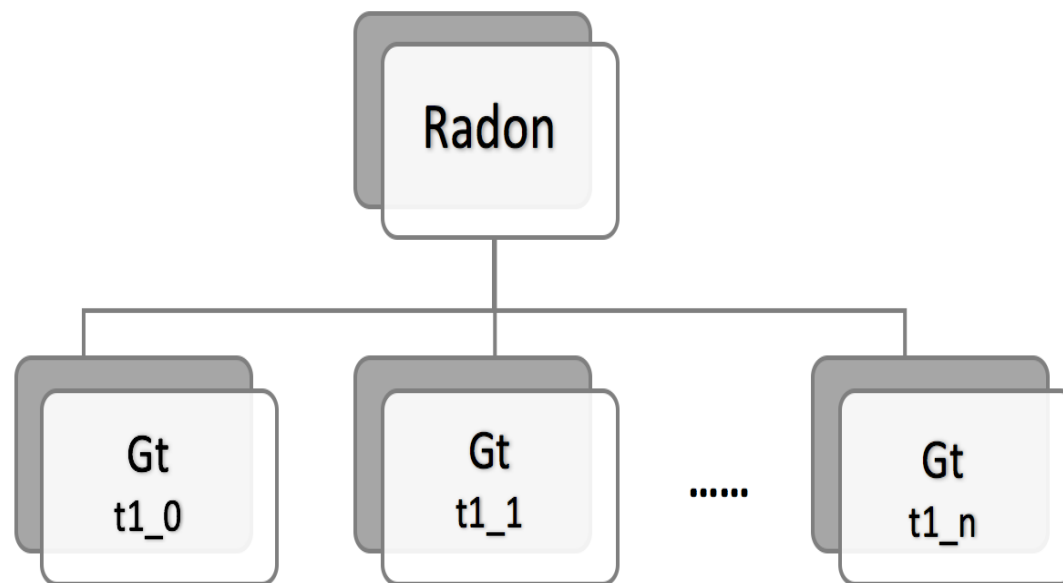
QINGCLOUD



RadonDB

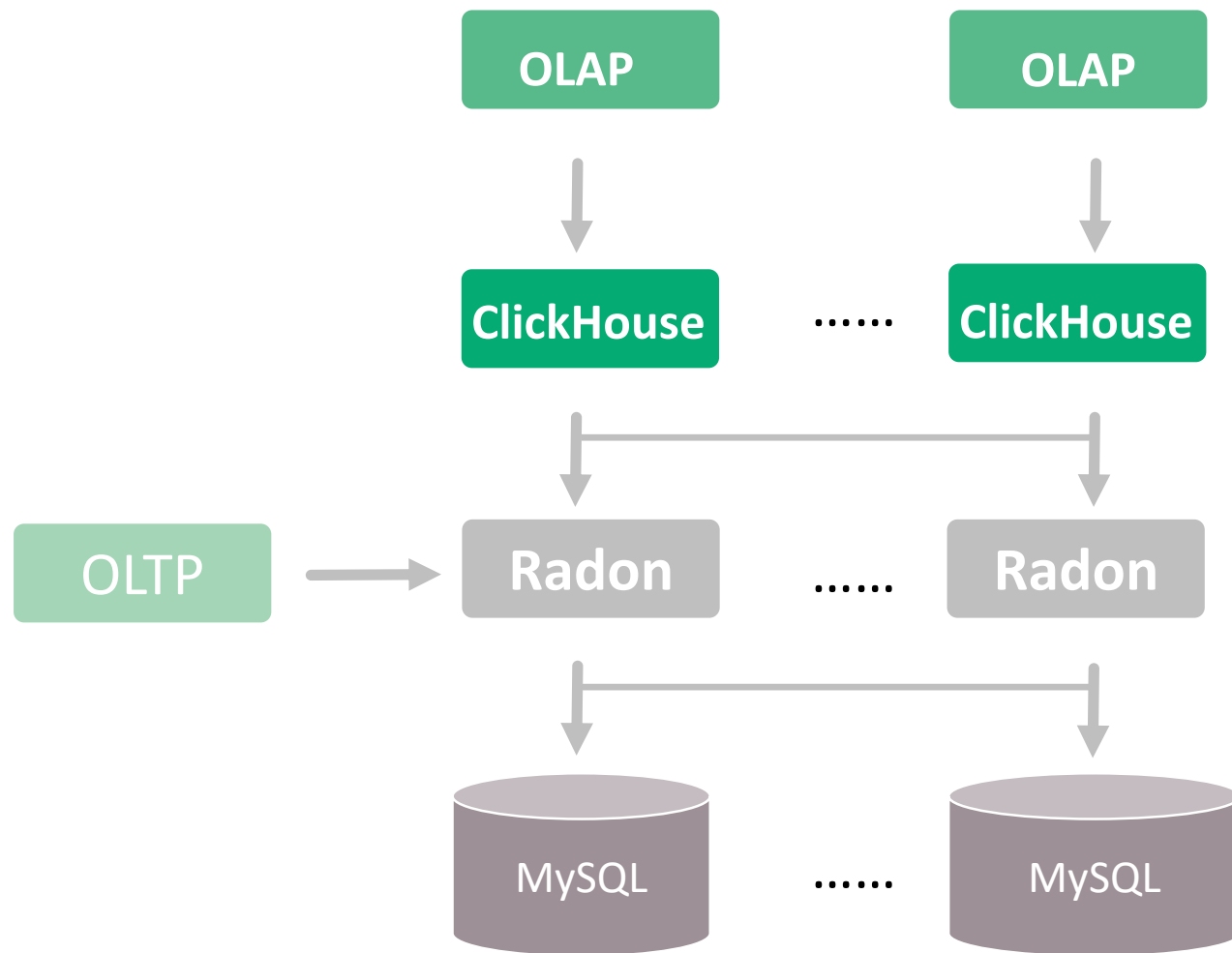
支持全局表join

- ▶ 实际数据库场景中，经常需要一些字典表：数据量小、变更频率低，可创建为全局表
- ▶ 全局表之间、全局表与分区表间join操作可根据路由下推存储节点执行，执行效率高，避免了跨库Join，执行结果在SQL节点汇总



OLAP = RadonDB+ClickHouse

- ▶ 分析性相关计算和复杂查询
- ▶ 向量化执行引擎和多重优化
- ▶ 未来和ClickHouse更深度融合



QINGCLOUD



RadonDB

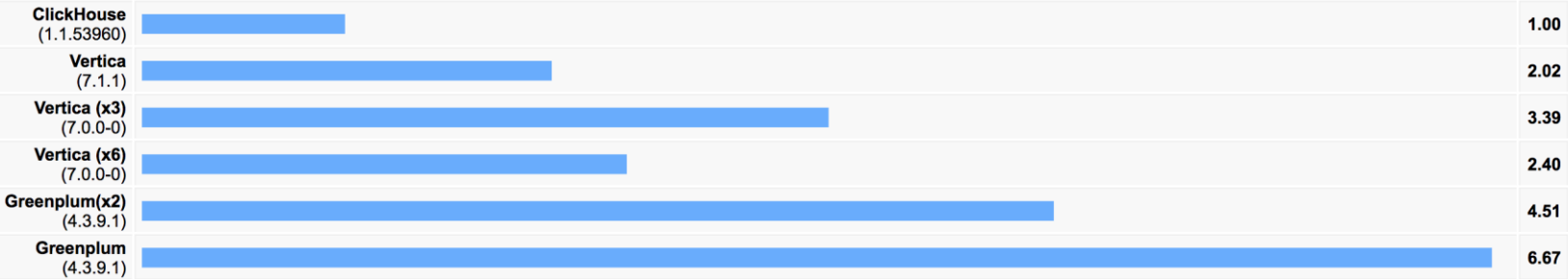
What is ClickHouse?

Compare: ClickHouse ClickHouse, new Vertica Vertica (x3) Vertica (x6) InfiniDB MonetDB Infobright Hive MySQL MemSQL Greenplum(x2) Greenplum

Dataset size: 10 mln. 100 mln. 1 bn.

Run number: first (cold cache) second third

Relative query processing time (lower is better):



Full results:

Query	ClickHouse (1.1.53960)	Vertica (7.1.1)	Vertica (x3) (7.0.0-0)	Vertica (x6) (7.0.0-0)	Greenplum(x2) (4.3.9.1)	Greenplum (4.3.9.1)
SELECT count() FROM hits	x7.73 (0.736 s.)	x3.12 (0.815 s.)	x34.97 (3.328 s.)	x23.15 (2.203 s.)	x186.00 (17.700 s.)	x304.85 (29.010 s.)
SELECT count() FROM hits WHERE AdvEngine	x3.12 (0.815 s.)	x4.92 (0.815 s.)	x17.40 (2.889 s.)	x13.26 (2.201 s.)	x55.06 (9.140 s.)	x126.08 (20.930 s.)
SELECT sum(AdvEngineID), count(), avg(Resol	x1.29 (2.111 s.)	x3.46 (3.667 s.)	x2.24 (3.669 s.)	x10.47 (17.130 s.)	x21.03 (34.410 s.)	
SELECT sum(UserID) FROM hits	x1.91 (4.059 s.)	x2.43 (5.163 s.)	x1.66 (3.512 s.)	x10.02 (21.250 s.)	x20.75 (44.010 s.)	
SELECT uniq(UserID) FROM hits	x1.83 (8.150 s.)	x2.41 (10.743 s.)	x1.68 (7.482 s.)	x3.85 (17.160 s.)	x8.28 (36.870 s.)	
SELECT uniq(SearchPhrase) FROM hits	x4.54 (26.119 s.)	x7.65 (44.021 s.)	x7.52 (43.298 s.)	x4.47 (25.700 s.)	x8.07 (46.440 s.)	
SELECT min(EventDate), max(EventDate) FRO	x1.77 (1.035 s.)	x5.21 (3.046 s.)	x3.91 (2.385 s.)	x25.25 (14.770 s.)	x49.95 (29.220 s.)	
SELECT AdvEngineID, count() FROM hits WHE	x2.45 (0.817 s.)	x8.97 (2.987 s.)	x6.34 (2.111 s.)	x35.44 (11.800 s.)	x61.80 (20.580 s.)	
SELECT RegionID, uniq(UserID) AS u FROM hil	x1.98 (13.108 s.)	x2.42 (16.039 s.)	x1.92 (12.751 s.)	x5.06 (33.570 s.)	x8.49 (56.290 s.)	
SELECT RegionID, sum(AdvEngineID), count() /	x5.53 (40.936 s.)	x3.24 (23.850 s.)	x2.26 (16.751 s.)	x8.33 (61.650 s.)	x10.70 (79.130 s.)	
SELECT MobilePhoneModel, uniq(UserID) AS u	x1.49 (4.735 s.)	x2.23 (7.084 s.)	x1.50 (4.789 s.)	x4.30 (13.690 s.)	x10.09 (32.110 s.)	
SELECT MobilePhone, MobilePhoneModel, unic	x1.23 (4.880 s.)	x2.37 (9.390 s.)	x1.52 (6.015 s.)	x3.65 (14.480 s.)	x8.49 (33.690 s.)	
SELECT SearchPhrase, count() AS c FROM hits	x3.75 (36.736 s.)	x5.08 (41.674 s.)	x5.22 (42.854 s.)	x3.12 (25.590 s.)	x5.46 (44.850 s.)	
SELECT SearchPhrase, uniq(UserID) AS u FRC	x6.31 (76.889 s.)	x11.35 (138.331 s.)	x4.13 (50.209 s.)	x2.52 (30.750 s.)	x5.18 (63.160 s.)	

<https://clickhouse.yandex/benchmark.html>

```

SELECT
  Carrier,
  c,
  c2,
  (c * 1000) / c2 AS c3
FROM
(
  SELECT
    Carrier,
    count(*) AS c
  FROM ontime
  WHERE (DepDelay > 10) AND (Year = 2007)
  GROUP BY Carrier
)
ANY INNER JOIN
(
  SELECT
    Carrier,
    count(*) AS c2
  FROM ontime
  WHERE Year = 2007
  GROUP BY Carrier
) USING (Carrier)
ORDER BY c3 DESC

```

**4核8G 1.5亿数据， 109
列， 美国航空飞行数据**

Carrier	c	c2	c3
EV	101796	286234	355.6390924907593
US	135987	485447	280.1273877477871
AA	176203	633857	277.98541311368336
MQ	145630	540494	269.43869867195565
AS	42830	160185	267.3783437899928
B6	50740	191450	265.0300339514233
UA	128174	490002	261.5785241692891
WN	296293	1168871	253.48648396615195
OH	59034	236032	250.11015455531452
CO	76662	323151	237.23274877688758
F9	23035	97760	235.62806873977087
YV	67905	294362	230.68534661403305
XE	99915	434773	229.8095787916913
FL	59460	263159	225.9470510223857
NW	90429	414526	218.15036933750838
OO	127426	597880	213.1297250284338
DL	93675	475889	196.8421207466447
9E	46948	258851	181.3707499681284
AQ	4299	46360	92.73080241587576
HA	2746	56175	48.88295505117935

20 rows in set. Elapsed: 12.740 sec. Processed 9.28 million rows, 128.02 MB (728.78 thousand rows/s., 10.05 MB/s.)




QINGCLOUD



RadonDB

xenon

 radondb / xenon

Unwatch ▾23

★ Unstar141

Fork35

<> Code

! Issues4

🔗 Pull requests0

📁 Projects0

📖 Wiki

📊 Insights

⚙ Settings

The MySQL Cluster Autopilot Management with GTID and Raft

Edit

mysql

high-availability

raft

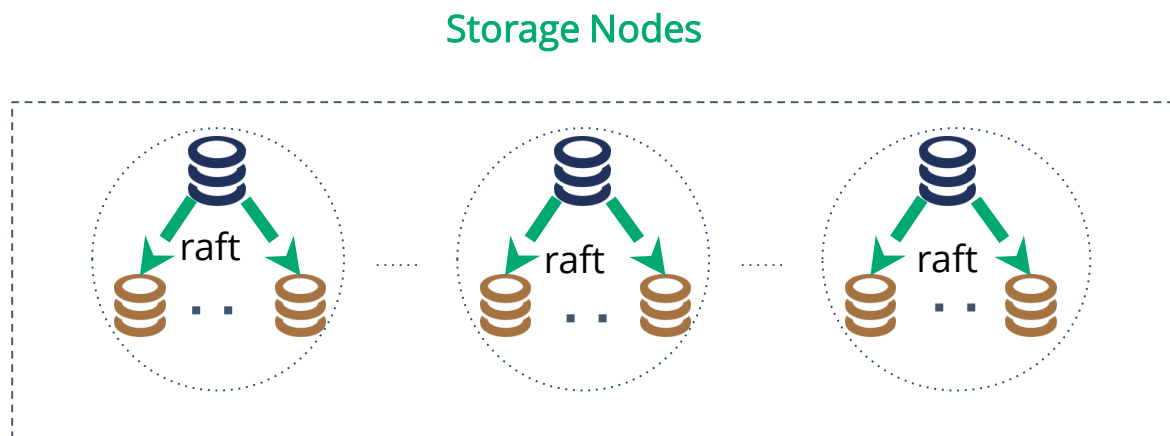
management-system

gtid

Manage topics

Storage Nodes

- ▶ 存储层由多个 node 组成
- ▶ 每个node 由多副本组成
- ▶ 每个副本为一个 **MySQL**
- ▶ 不仅存储还有计算能力

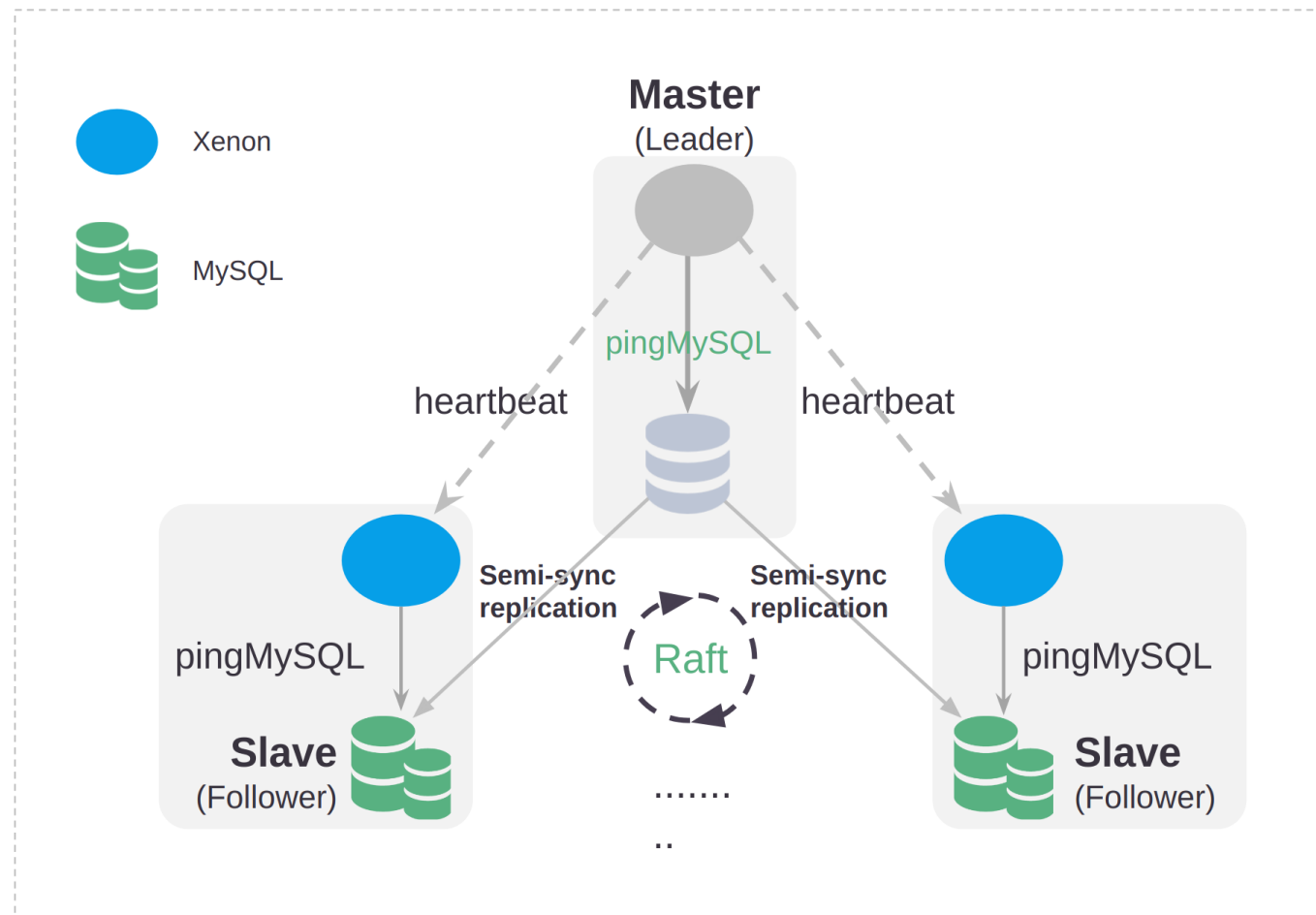


QINGCLOUD

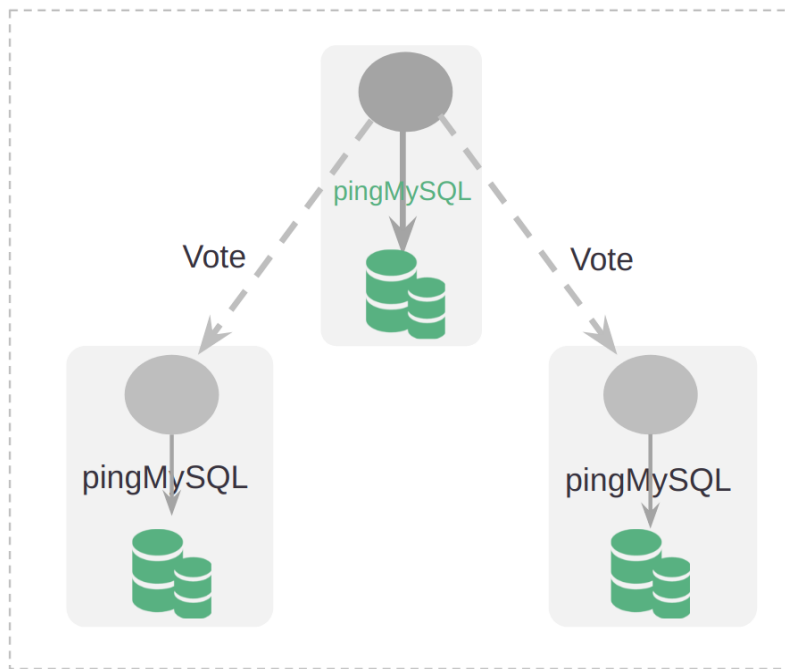


RadonDB

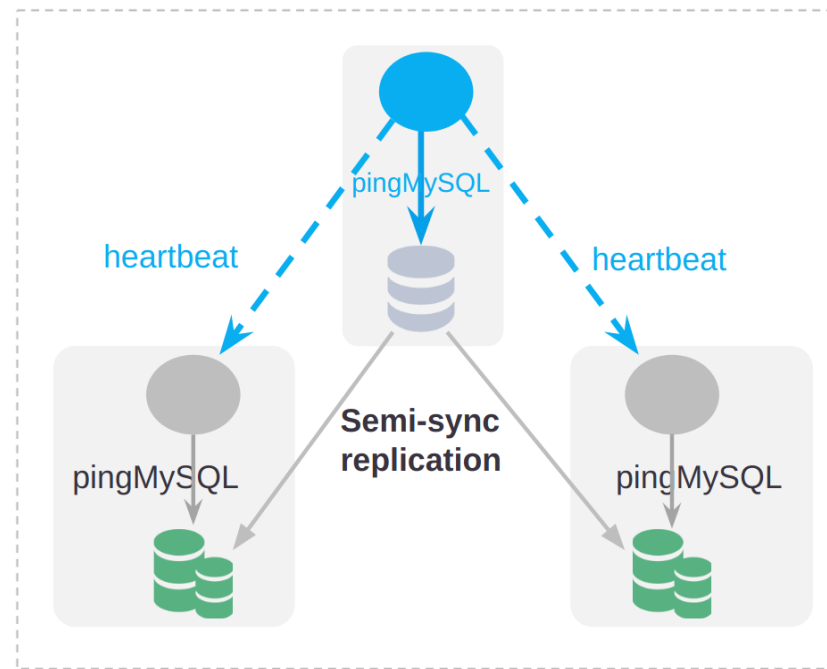
Xenon架构



初始状态

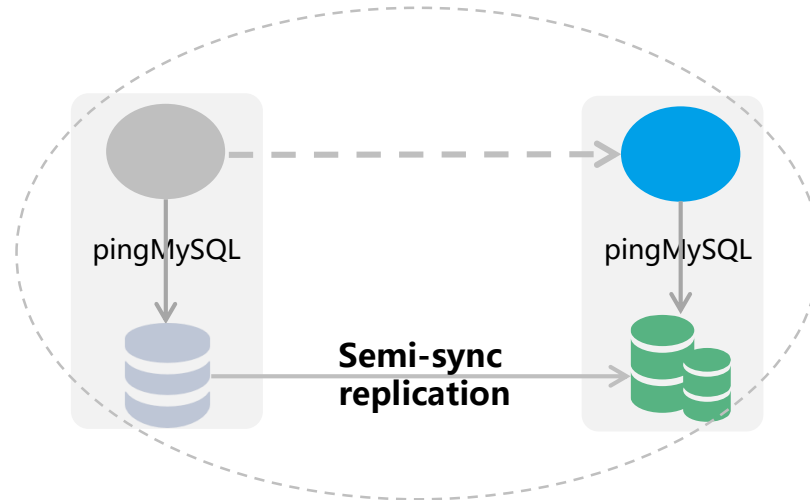
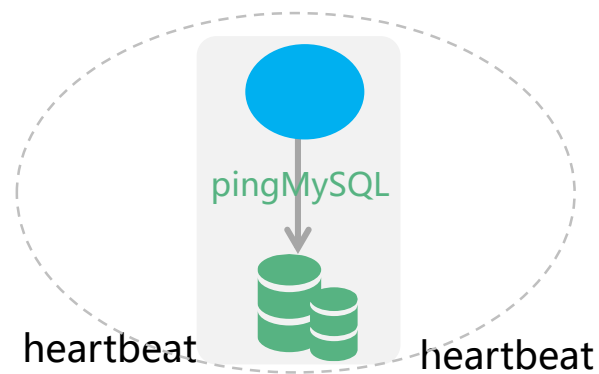
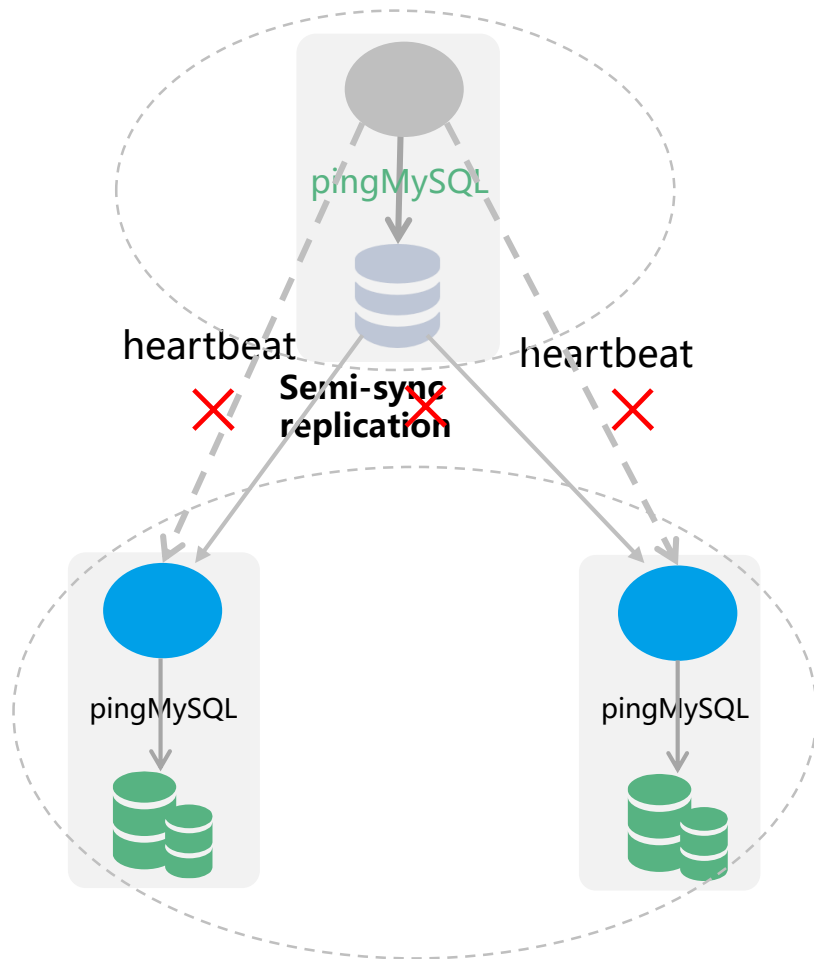


发起选举

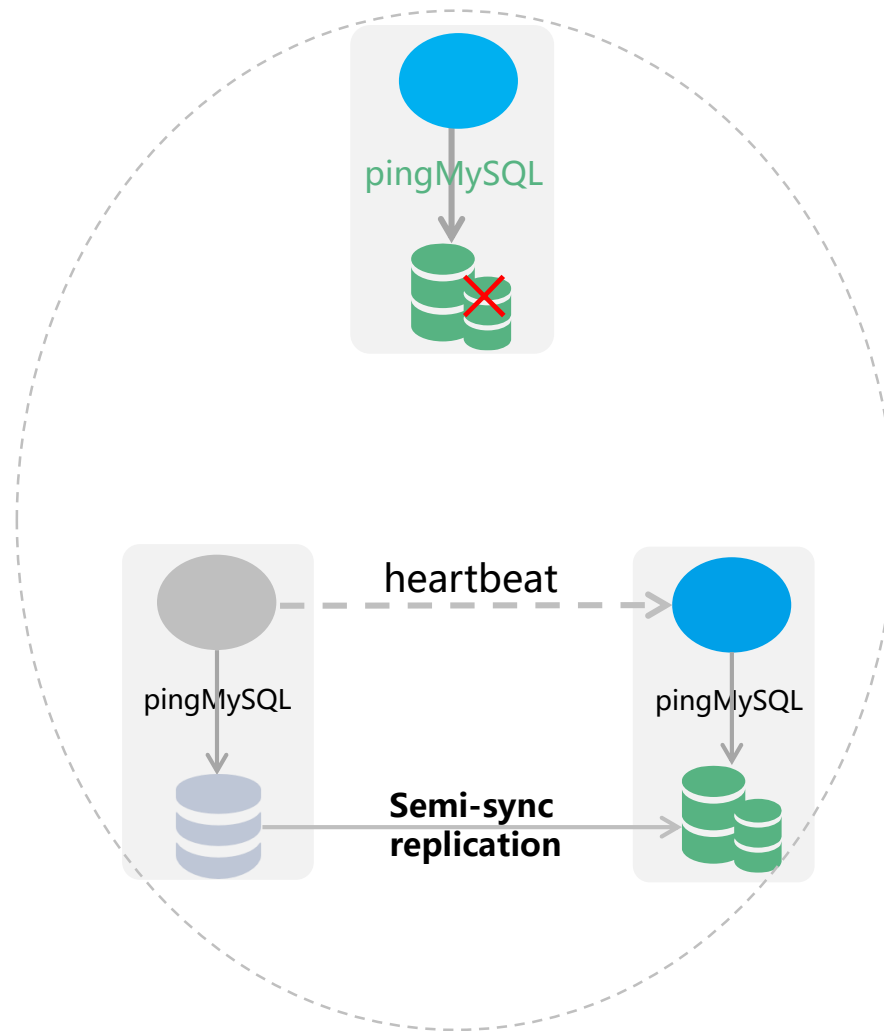
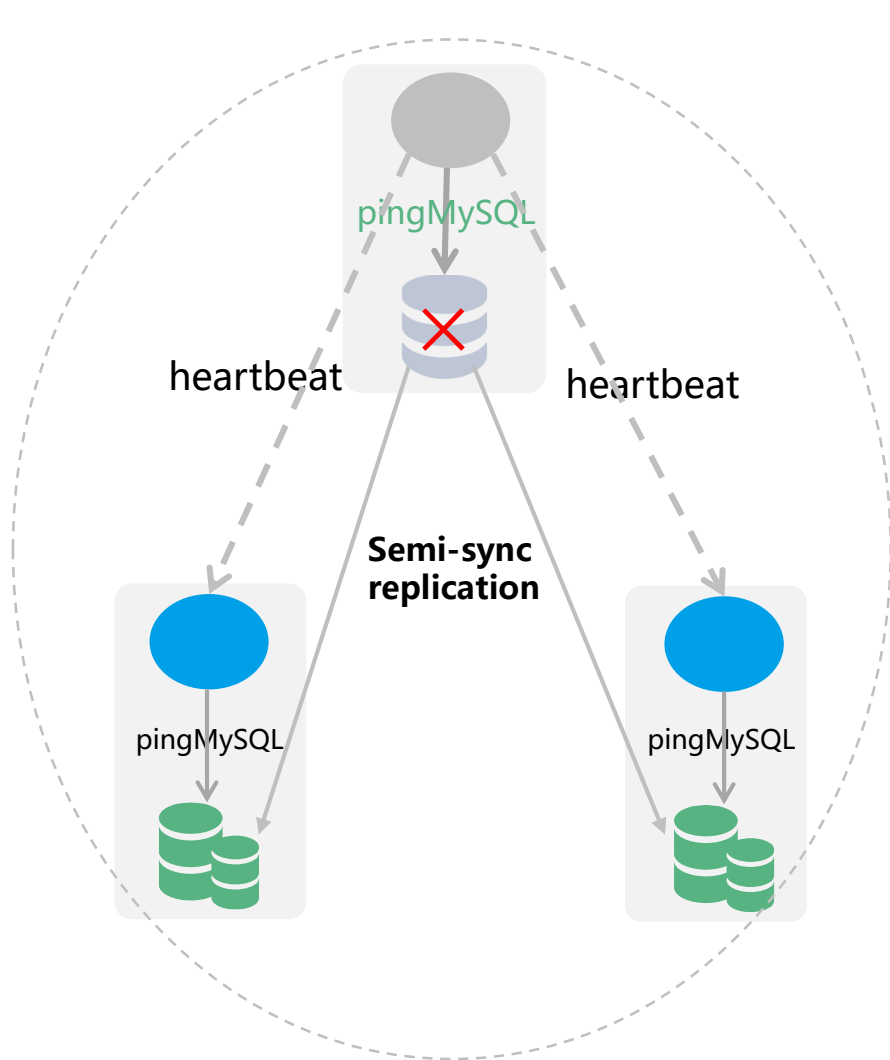


当选为Leader

网络隔离

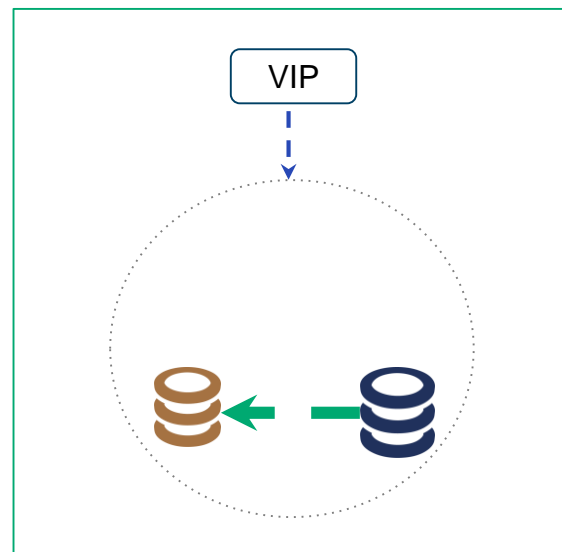
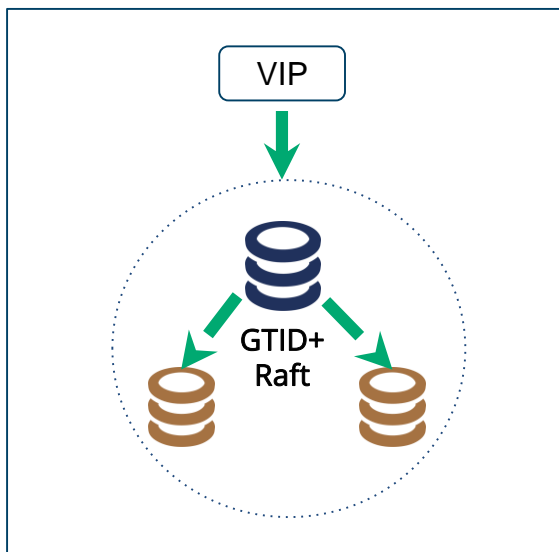


Leader 故障



高可用

- ▶ GTID 作为 Raft Log Index
- ▶ Raft 协议选主、Log 并行复制
- ▶ 主副本故障秒级切换即可服务
- ▶ 强 Semi-Sync 确保事务不丢失
- ▶ 单副本故障可快速流式重建
- ▶ 无中心化，可跨机房部署



Raft + MySQL = Raft 选主 + GTID 并行复制 + 强 Semi-Sync
数据强一致、切换零丢失



QINGCLOUD



RadonDB

性能

sysbench: 16表, 512线程, 随机写, 5000万条数据

	Transaction Per Second(TPS)	Response Time(avg)	规格
RadonDB (1SQL节点, 4 存储节点)	26,589	20ms	4 存储节点(16C64G超高性能主机) sync_binlog=1 innodb_flush_log_at_trx_commit=1
单机 MySQL (QingCloud RDB)	9,346	73ms	RDB(16C64G超高性能主机) sync_binlog=1 innodb_flush_log_at_trx_commit=1



QINGCLOUD



RadonDB

审计日志

- ▶ 支持多种审计模式
- ▶ 可定位慢查询等

```
type event struct {  
▶ Start      time.Time    `json:"start"`    // Time the query was start.  
▶ End        time.Time    `json:"end"`      // Time the query was end.  
▶ Cost       time.Duration `json:"cost"`     // Cost.  
▶ User       string       `json:"user"`     // User.  
▶ UserHost   string       `json:"user_host"` // User and host combination.  
▶ ThreadID   uint32       `json:"thread_id"` // Thread id.  
▶ CommandType string       `json:"command_type"` // Type of command.  
▶ Argument   string       `json:"argument"`  // Full query.  
▶ QueryRows  uint64       `json:"query_rows"` // Query rows.  
}
```



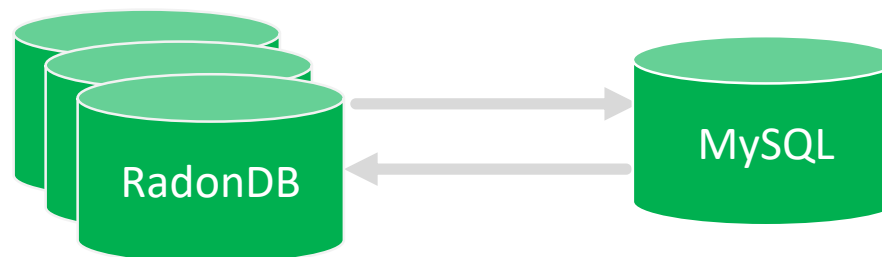
QINGCLOUD



RadonDB

Backup & restore

- ▶ xelabs/go-mydumper
- ▶ 批量并行流式导出
- ▶ 批量并行导入
- ▶ 在线数据同步工具



监控信息

- ▶ 全链路监控
- ▶ mysql> show processlist;
- ▶ mysql> show txnz;
- ▶ mysql> show queryz;

```
mysql> show txnz;
```

TxnID	Start	Duration	XaState	TxnState
157752	20171127133617.898	2.146426ms	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157755	20171127133617.898	2.056694ms	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157757	20171127133617.898	1.852888ms	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157758	20171127133617.899	1.280103ms	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157759	20171127133617.899	1.168348ms	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157760	20171127133617.899	1.12425ms	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157761	20171127133617.899	989.395µs	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157762	20171127133617.899	989.942µs	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157763	20171127133617.899	535.606µs	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157764	20171127133617.899	540.083µs	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157765	20171127133617.900	304.503µs	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157766	20171127133617.900	157.271µs	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157767	20171127133617.900	152.204µs	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157768	20171127133617.900	57.664µs	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157769	20171127133617.900	50.243µs	XASTATE_START_FINISHED	TXNSTATE_EXECUTINGTWOPC
157770	20171127133617.900	26.454µs	XASTATE_START	TXNSTATE_LIVE

```
16 rows in set (0.00 sec)
```

```
mysql> show queryz;
```

ConnID	Host	Start	Duration	Query
653	192.168.0.2:3306	20171127133636.572	26.100427ms	insert into sbtest.benchyou13_0003(k, c, pad, id) values (1656722355343024427, '65332289937-13041298506-25618055297-41535934916- [TRUNCATED]
662	192.168.0.2:3306	20171127133636.572	25.941914ms	insert into sbtest.benchyou7_0006(k, c, pad, id) values (412206200411920065, '11464577720-87417918501-83129869594-25483513101-60 [TRUNCATED]
659	192.168.0.2:3306	20171127133636.573	25.469511ms	insert into sbtest.benchyou11_0007(k, c, pad, id) values (2923743091818020252, '82152288392-36903295031-65430756464-26180168569- [TRUNCATED]
655	192.168.0.2:3306	20171127133636.574	24.322072ms	insert into sbtest.benchyou12_0000(k, c, pad, id) values (7094417003669075702, '64976631780-66856105580-06242610079-23601332731- [TRUNCATED]
658	192.168.0.2:3306	20171127133636.574	24.178377ms	insert into sbtest.benchyou5_0007(k, c, pad, id) values (2537338909164074961, '90808405806-91930963083-53042963931-08744061744-7 [TRUNCATED]
649	192.168.0.2:3306	20171127133636.574	23.933553ms	insert into sbtest.benchyou14_0009(k, c, pad, id) values (7668724556483513943, '51698352524-41749630204-12325428781-39818670108- [TRUNCATED]
654	192.168.0.2:3306	20171127133636.575	23.307209ms	insert into sbtest.benchyou13_0002(k, c, pad, id) values (9031333885746166268, '04682641064-62753344570-36190143839-67814638490- [TRUNCATED]
663	192.168.0.2:3306	20171127133636.575	22.935888ms	insert into sbtest.benchyou1_0000(k, c, pad, id) values (4624277464688657385, '52978656463-13518003014-01750792432-45981516887-2 [TRUNCATED]
660	192.168.0.2:3306	20171127133636.575	22.68368ms	insert into sbtest.benchyou11_0006(k, c, pad, id) values (5990639287556878520, '42819656260-38683758739-92836595066-06782629878- [TRUNCATED]
664	192.168.0.2:3306	20171127133636.577	21.289224ms	insert into sbtest.benchyou3_0002(k, c, pad, id) values (8413237746760191843, '07416097870-55681347132-70713893811-97510616773-1 [TRUNCATED]
650	192.168.0.2:3306	20171127133636.577	21.287309ms	insert into sbtest.benchyou2_0009(k, c, pad, id) values (4564716827037085640, '47054643499-92772864590-21825206587-60244686970-1 [TRUNCATED]
657	192.168.0.2:3306	20171127133636.577	20.884917ms	insert into sbtest.benchyou14_0002(k, c, pad, id) values (1114671067020431775, '46636743337-32237471368-43862774993-70427218508- [TRUNCATED]
651	192.168.0.2:3306	20171127133636.578	19.685514ms	insert into sbtest.benchyou10_0009(k, c, pad, id) values (1849090697326484382, '01634400882-46927678250-93224930378-27748921244- [TRUNCATED]
661	192.168.0.2:3306	20171127133636.581	17.445214ms	insert into sbtest.benchyou1_0001(k, c, pad, id) values (6848070351960303464, '89399961063-09992892450-77700841814-39274094320-9 [TRUNCATED]
652	192.168.0.2:3306	20171127133636.582	15.633855ms	insert into sbtest.benchyou7_0009(k, c, pad, id) values (5572355138304685994, '75736956542-93823728904-76083806301-09336795205-8 [TRUNCATED]
656	192.168.0.2:3306	20171127133636.596	2.515133ms	insert into sbtest.benchyou4_0000(k, c, pad, id) values (3884875362167629873, '33384277147-53242667960-61971971190-96128522627-7 [TRUNCATED]

```
16 rows in set (0.00 sec)
```



QINGCLOUD



RadonDB

监控图形化展示

- ▶ Radon代码程序中定制需要的Metric
- ▶ 开源时序数据库Prometheus作为监控和性能指标信息存储方案
- ▶ 使用Grafana作为可视化组件进行展示



展望

- ▶ MyNewSQL 刚刚开始
- ▶ Hybrid Transactional/Analytical Processing



QINGCLOUD



RadonDB

2018年5月 新一代分布式关系型数据库 RadonDB 正式开源

如果您希望马上参与其中, 请访问我们的 GitHub 页面, 获取源代码、构建项目、下载最新版本, 甚至开始贡献您自己的力量:



项目网址
radondb.io



代码地址
<https://github.com/radondb>

我们期待与大家一起构建分布式数据库的未来。



QINGCLOUD



RadonDB

关于「3306π」社区

围绕MySQL核心技术，将互联网行业中最重要
的数据化解决方案带到传统行业中
囊括**其他开源技术**，redis、MongoDB、Hbase、
Hadoop、ElasticSearch、Storm、Spark等
在全面互联网化的大趋势下，将互联网新鲜的核心技
术理念带到传统行业里，构建良好交流互动环境
分享干货知识，即便是赞助商，也要求如此，拒绝
放水

「3306 π 」社区，欢迎您的加入



社区公众号



社区QQ群



Thank you.

Andy@yunify.com



QingCloud-IaaS



青云QingCloud

www.qingcloud.com