PingCAP

# TiDB 在竞技世界从 1-N 的应用实践

## --从选型到落地，助力构建高性能的数据底座

魏建强

# 关于竞技世界

竞技世界（北京）网络技术有限公司成立于 2007 年，主要从事网络游戏研发、运营等业务。旗下休闲竞技游戏平台“JJ斗地主”，为用户提供包括斗地主在内的数十款游戏，玩法多样，比赛丰富，以竞技为核心、坚持绿色运营，至今注册用户数已超 5 亿。

竞技世界在作为大众棋牌竞技化的推动者的同时，也致力于发展绿色休闲游戏，并积极涉足自研、发行、投资等领域，自主研发了《热血街篮》《曙光英雄》《我自为道》等多款休闲手游。



# 目录

- 为什么选择TiDB
- 如何从0到1应用TiDB
- 使用TiDB的收益
- 遇到的一些问题

# MySQL + MyCAT的痛点

**MySQL存储瓶颈**

底层资源架构的限制，存储限制和无法实现弹性伸缩

**MySQL性能瓶颈**

超大表并发能力有限
读写分离从库延迟，

**MySQL高可用**

依赖MHA，存在误切及故障影响业务的读写

**分库分表，聚合难度大**

聚合查询难度大，非分片查询效率低，事务支持能力有限

**MySQL推荐场景OLTP**

分析统计类查询效率低，甚至查询不出结果

**MyCAT扩缩容分片复杂**

分片键选择复杂
分片扩容或缩容难度大

# 常见解决方案


- 按照业务库进行拆分
- 按照业务数据类型拆分,例如：字典、账单、状态
- 按照业务模型进行拆分
- 数据孤岛问题


- 聚合难度大、非分片查询效率低
- 增加或减少分片难度大
- 运维成本高


- 针对复杂分析查询需求的数据下沉到CK、Doris、大数据集群、数仓
- 针对数据存储需求数据归档到大数据集群

# 选型的四项准则

以**架构演进**的视角规划

以技术生态和**趋势发展**为参考


以**业务场景**驱动
需要业务侧紧密配合

**多元化**，拒绝ALL IN ONE

# 为什么选择TiDB？

## 01 稳定性和扩展性

1、数据库一致性
2、原生高可用和容灾能力
3、弹性扩缩容,业务无感
4、支持事务

## 02 业务需求

1、海量数据存储
2、高性能和高并发
3、业务改造和接入成本低
4、实时 HTAP

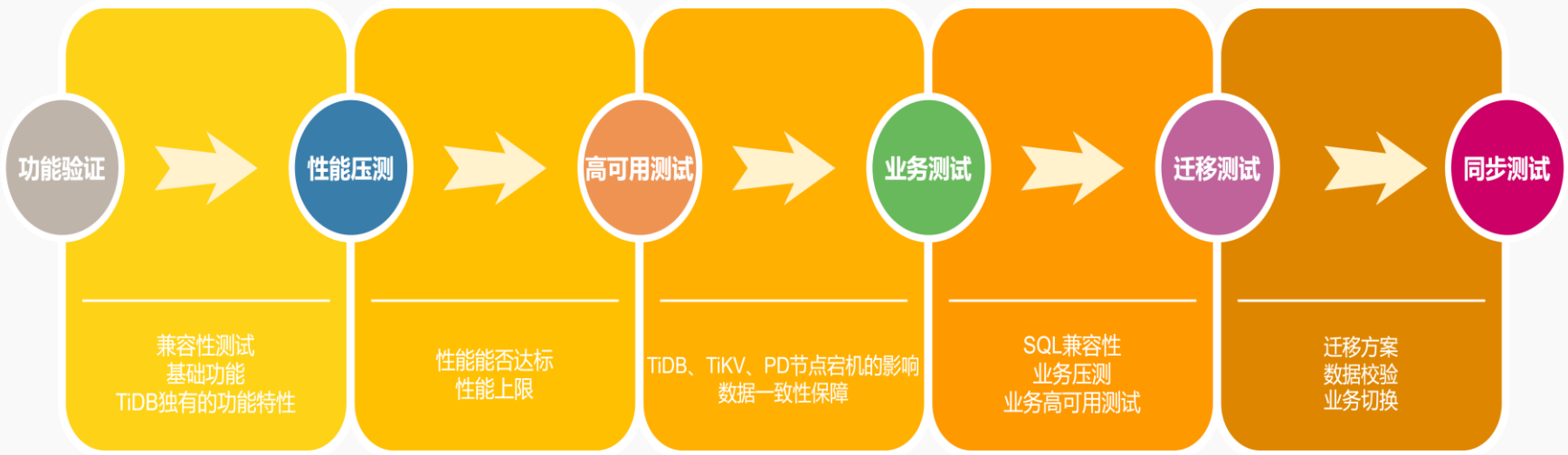
## 03 运维管理

1、简化运维管理
2、兼容MySQL协议和MySQL生态

## 04 成熟度

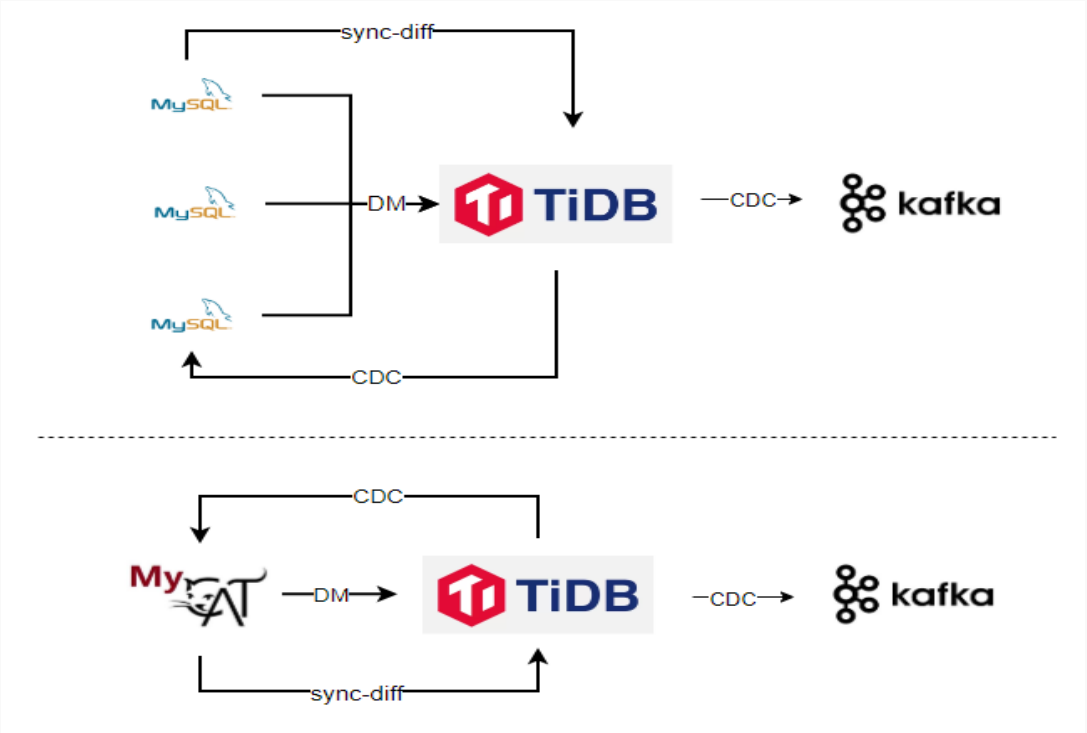
1、开源产品
2、周边生态丰富
3、社区成熟

# 选型阶段的验证性测试流程



# MySQL到TiDB迁移

- SQL可兼容
- 迁移可回退
- 性能不能差



# 迁移后--性能提升

| 高频复杂SQL查询性能对比 | | | |
|---|---|---|---|
| 查询时间 | MySQL | TiDB | TiFlash |
| SQL1 | 5.44s | 0.14s | 0.42s |
| SQL2 | 16.79s | 0.23s | 0.26s |
| SQL3 | 1 min 30.45s | 1.29s | 1.05s |
| SQL4 | 3 min 58.43s | 1.01s | 0.42s |
| SQL5 | 5 min 54.58s | 3.57s | 1.90s |

在MySQL中执行在30s以上的SQL语句：

1、95%的SQL语句查询时间缩短在1s内

2、高频复杂SQL基本控制在3s内

3、75%的SQL耗时都在300ms内

4、查询时间跨度更大，近半年 -> 近2年

# TiDB应用场景实践

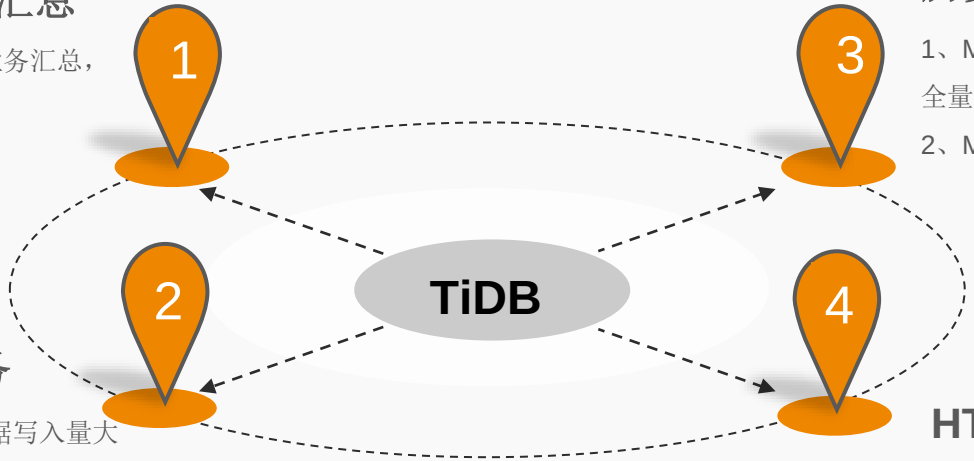
## 分库分表汇总

1、上游分库分表集群业务汇总，满足聚合查询需求

## 历史数据归档查询

1、MySQL历史数据归档，TiDB保留全量数据

2、MySQL存储瓶颈和汇总查询慢




## 高IO业务

1、账单类存储，数据写入量大

2、流转需求

3、关联查询

## HTAP

1、同时处理OLAP和OLTP，解决繁琐的ETL过程

# TiDB当前规模和收益

9套集群，115+节点，目前主要版本为5.0和6.5

研发侧

1、使用和接入方便，数据存储更多、聚合查询更快、数据统计分析更实时，研发更专注业务创新
2、抗住截止目前10+次的物理机异常宕机或服务异常事故的考验，TiDB集群所承载的业务无不可用的影响；

1、兼容MySQL生态、集群稳定性和数据一致性有保障
2、弹性扩缩容40+次，主机迁移5+次，运维管理效率提升了10多倍，极大的降低的运维成本

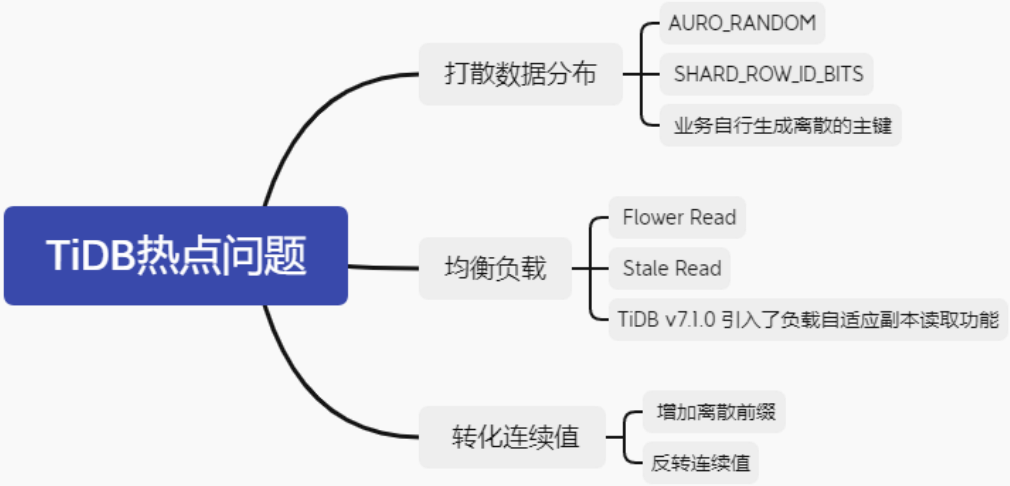
运维侧

# 遇到的典型问题

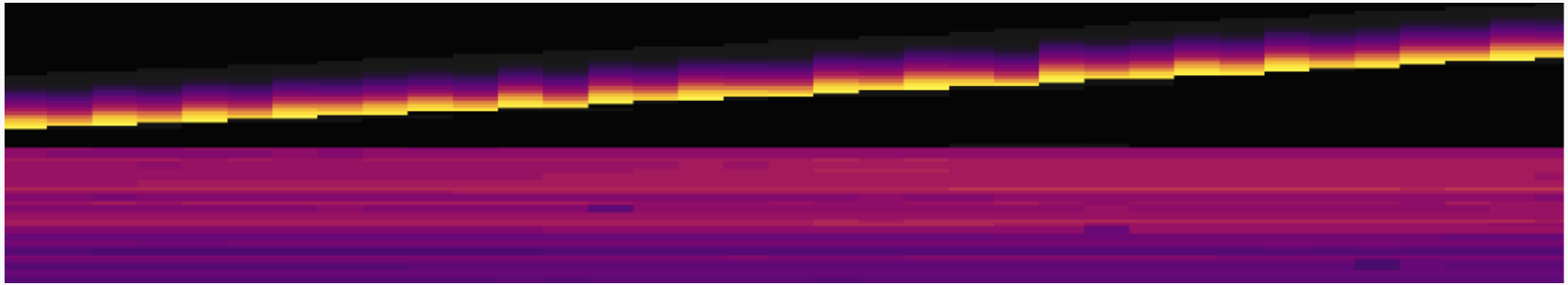
## 问题1：热点问题

1、从MySQL迁移到TiDB 的业务,迁移前会使用自增主键,将随机写转为顺序写提高性能。在写入较大的数据,会造成写入热点,对于这部分写入就会退化成单机的写入性能，未能利用分布式读写扩展的优势

2、很多账单类型的业务,经常需要按照时间的维度查询,自然需要对时间字段创建索引,可能产生索引热点,导致写入的吞吐受到影响



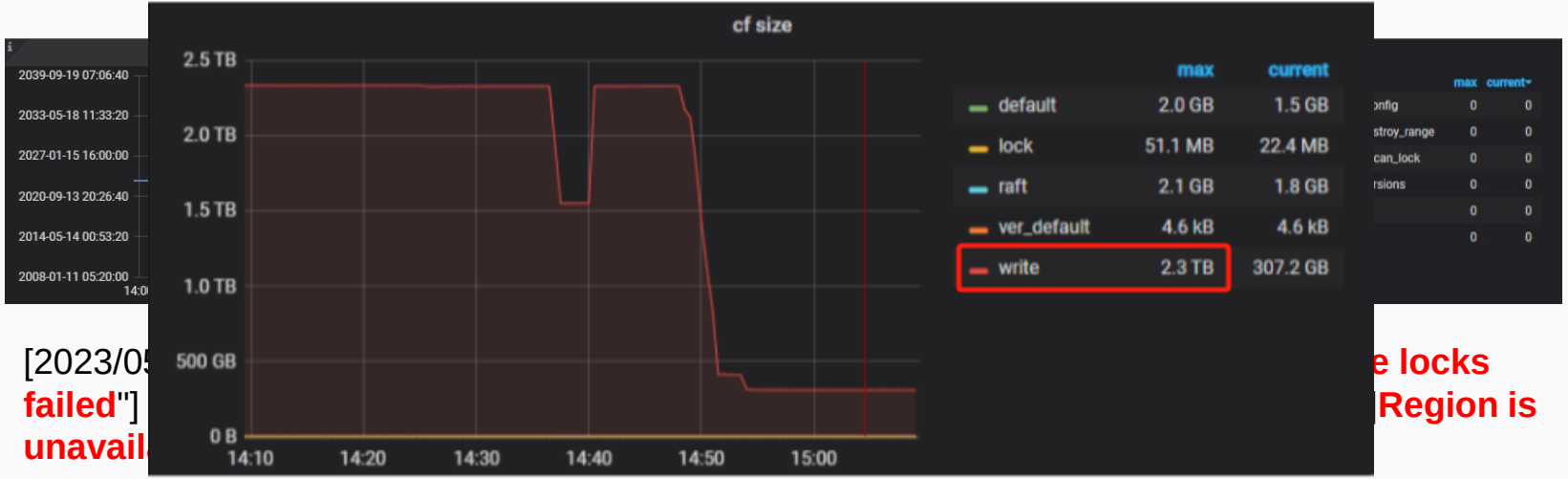
优化前：



优化后：

## 问题2：GC不工作

**主要现象**：(1) safe point 长时间不推进　(2) 删除数据后磁盘空间一直没有回收




[2023/05 e locks
failed"] Region is
unavail

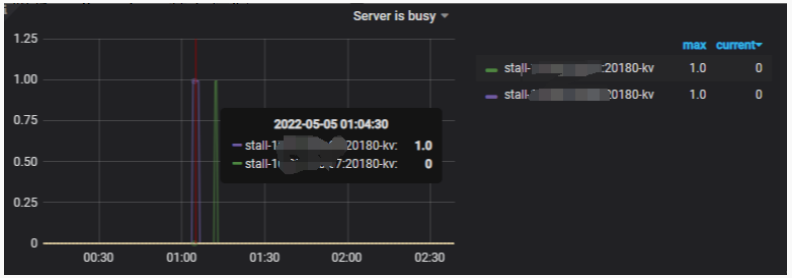
**解决问题**：排查非3个副本的region和空region,发现有一个没有leader的空region;尝试删除该region和清理pd中该region信息无效，最后重建region，重启相关的TiKV节点

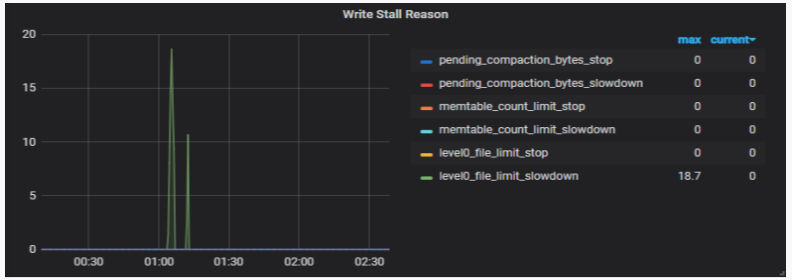
# 问题3：Write Stalls

Write Stalls 是 rocksdb 的一种流控机制，当 flush/compaction 赶不上 write rate 的速度时，rocksdb 会降低 write rate，甚至停写

Write Stalls常见的三种场景：

- Memtable 文件数量过多
- L0 层 SST 文件数量过多
- L1 ~ Ln 层待 Compaction 的 SST 文件的大小过大



**解决问题：**

1、分析磁盘 IO 压力太大导致的
如果磁盘 IO 压力和 CPU 压力不大，适当调整TiKV并发刷盘参数可以解决Write Stalls的问题。
如果磁盘 IO 压力和 CPU 压力大，说明Write Stalls的本质原因是磁盘 IO 跟不上导致的，调整某些参数阀值只能缓解，无法从根本上解决问题，可以考虑业务端限流或者集群扩容

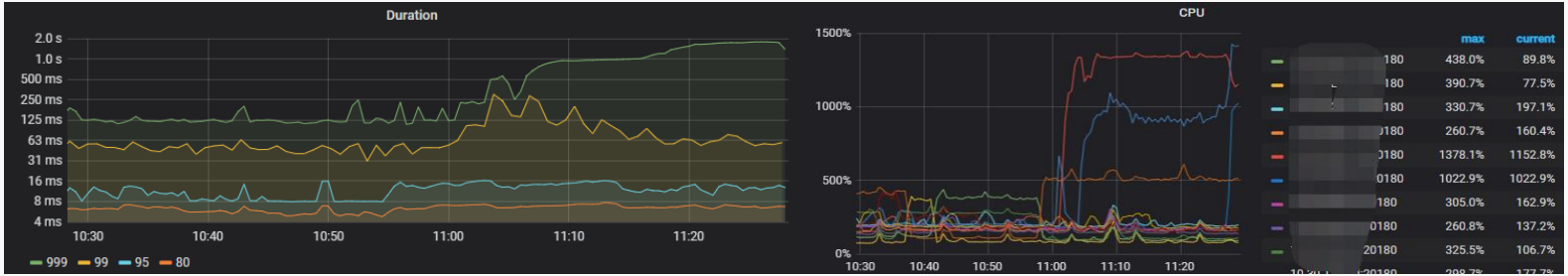
2、TiKV并发刷盘参数太小导致的

rocksdb.defaultcf.write-buffer-size
rocksdb.defaultcf.max-write-buffer-number
rocksdb.max-background-flushes
rocksdb.defaultcf.soft-pending-compaction-bytes-limit
rocksdb.writecf.soft-pending-compaction-bytes-limit
rocksdb.defaultcf.hard-pending-compaction-bytes-limit
rocksdb.writecf.hard-pending-compaction-bytes-limit

rocksdb.defaultcf.level0-slowdown-writes-trigger
rocksdb.writecf.level0-slowdown-writes-trigger
rocksdb.lockcf.level0-slowdown-writes-trigger
rocksdb.defaultcf.level0-stop-writes-trigger
rocksdb.writecf.level0-stop-writes-trigger
rocksdb.lockcf.level0-stop-writes-trigger

# 问题4：隐式转化

**主要现象**：(1) 延迟增加　(2) 某些TiKV节点CPU使用率增大




```
UPDATE `sms_order_xxxx` SET `report_code` = 1 WHERE `mt_msg_id` =
02303081108280371760;
```

| 执行计划 ID ⓘ | 累计耗时 ⓘ | 平均耗时 ⓘ | 执行次数 ⓘ | 平均内存 ⓘ |
|---|---|---|---|---|
| c02b766b22f0f55ae3e61953a788c3900b096aa... | 169.8 min | 1.1 s | 9.5 K | 11.4 KiB |
| d6b96b333924d9611037ebf37558cd8ae65907... | 4.2 s | 5.6 ms | 747 | 30.9 KiB |

PingCAP

# THANK YOU.