分库分表已成为过去式，使用分布式数据库才是未来

> 转载至我的博客 [www.infrastack.cn](http://cxyroad.com/ "https://www.infrastack.cn/?p=70")，公众号：架构成长指南

当我们使用 Mysql数据库到达一定量级以后，性能就会逐步下降，而解决此类问题，常用的手段就是引入数据库中间件进行分库分表处理，比如使用`Mycat`、`ShadingShpere`、`tddl`，但是这种都是过去式了，现在使用分布式数据库可以避免分库分表

为什么不建议分库分表呢?

分库分表以后，会面临以下问题

* 分页问题，例如：使用传统写法，随着页数过大性能会急剧下降
* 分布式事务问题
* 数据迁移问题，例如：需要把现有数据通过分配算法导入到所有的分库中
* 数据扩容问题，分库分表的数据总有一天也会到达极限，需要增大分片
* 开发模式变化，比如在请求数据时，需要带分片键，否则就会导致所有节点执行
* 跨库跨表查询问题
* 业务需要进行一定取舍，由于分库分表的局限性，有些场景下需要业务进行取舍

以上只是列举了一部分问题，为了避免这些问题，可以使用分布式数据库TiDB来处理

TiDB介绍

[TiDB](http://cxyroad.com/ "https://github.com/pingcap/tidb") 是 [PingCAP](http://cxyroad.com/ "https://pingcap.com/about-cn/") 公司研发的一款开源分布式关系型数据库，从 2015年 9 月开源，至今已经有9 年时间，可以说已经非常成熟，它是一款同时支持OLTP（在线事务处理）和 OLAP（在线分析处理）的融合型分布式数据库产品，具备水平扩缩容，金融级高可用、实时 HTAP（Hybrid Transactional and Analytical Processing）、云原生的分布式数据库，兼容 MySQL 5.7 协议和 MySQL 生态等重要特性

，它适合高可用、强一致要求较高、数据规模较大等各种应用场景。

核心特性

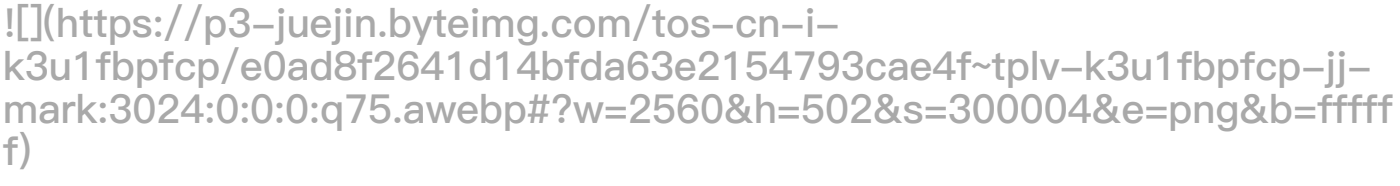
* 金融级高可用
* 在线水平扩容或者缩容，并且存算分离
* 云原生的分布式数据库，支持部署在公有云，私有云，混合云中
* 实时HTAP，提供TIKV行存储引擎和TiFlash列存储引擎
* 兼容MySQL协议和MySQL生态
* 分布式事务强一致性
* 从 MySQL 无缝切换到 TiDB，几乎无需修改代码，迁移成本极低
* PD在分布式理论CAP方面满足CP，是强一致性的

应用场景

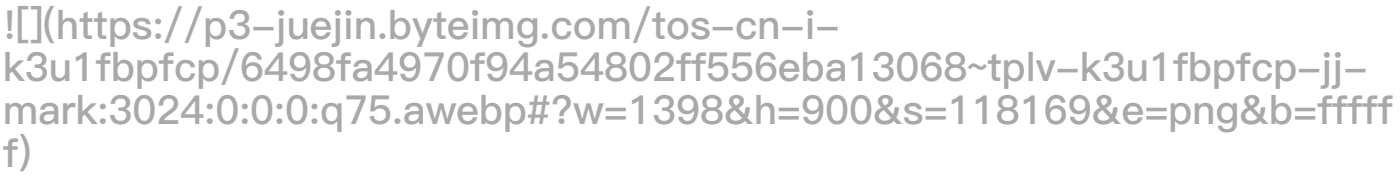
* 对数据一致性及高可靠、系统高可用、可扩展性、容灾要求较高的金融行业属性的场景
* 对存储容量、可扩展性、并发要求较高的海量数据及高并发的OLTP场景
* 数据汇聚、二次加工处理的场景

案例

TiDB 有**1500** 多家不同行业的企业应用在了生产环境，以下是一些有代表性企业，要想查看更多案例，可以访问TiDB 官网查询



系统架构



TIDB Server

SQL 层，对外暴露 MySQL 协议的连接 endpoint，负责接收SQL请求，处理SQL相关的逻辑，并通过PD找到存储计算所需数据的TiKV地址，与TiKV交互获取数据，最终返回结果。TiDB Server 是无状态的，其本身并不存储数据，只负责计算，可以无限水平扩展，可以通过负载均衡组件（LVS、HAProxy或F5）对外提供统一的接入地址，客户端的连接可以均匀地分摊在多个 TiDB 实例上以达到负载均衡的效果。

PD Server

整个集群的管理模块，其主要工作有三个：

1. 存储集群的元信息（某个Key存储在那个TiKV节点）；
2. 对TiKV集群进行调度和负载均衡、Leader选举；
3. 分配全局唯一且递增的事务ID。

PD 是一个集群，需要部署奇数个节点，一般线上推荐至少部署3个节点。PD在选举的过程中无法对外提供服务，这个时间大约是3秒。

TIKV Server

TiDB 现在同时支持OLTP 和 OLAP，而TiKV负责存储OLTP数据，从外部看TiKV是一个分布式的提供事务的Key-Value存储引擎。存储数据的基本单位是Region，每个Region负责存储一个Key Range（从StartKey到EndKey的左闭右开区间）的数据，每个TiKV节点会负责多个Region。

TiKV如何做到数据不丢失的?

简单理解，就是把数据复制到多台机器上，这样一个节点down 机，其他节点上的副本还能继续提供服务；复杂理解，需要这个数据可靠并且高效复制到其他节点，并且能处理副本失效的情况，那怎么做呢，就是使用 `Raft`一致性算法

Region 与副本之间通过 Raft 协议来维持数据一致性，任何写请求都只能在 Leader 上写入，并且需要写入多数副本后（默认配置为 3 副本，即所有请求必须至少写入两个副本成功）才会返回客户端写入成功。

分布式事务支持

TiKV 支持分布式事务，我们可以一次性写入多个 key-value 而不必关心这些 key-value 是否处于同一个数据切片 (Region) 上，TiKV 的分布式事务参考了 Google 在 BigTable 中使用的事务模型[Percolator](http://cxyroad.com/”https://research.google.com/pubs/pub36726.html”)，具体可以访问论文了解

与MySQL的对比

支持的特性

* 支持分布式事务，原理是基于Google Percolator，Percolator是基于Bigtable的，所以数据结构直接使用了Bigtable的Tablet。详情可参考[zhuanlan.zhihu.com/p/39896539](http://cxyroad.com/”https://zhuanlan.zhihu.com/p/39896539”)
* 支持锁，TIDB是乐观锁 +MVCC ，MySQL是悲观锁+MVCC，要注意TIDB执行Update、Insert、Delete时不会检查冲突，只有在提交时才会检查写写冲突，所以在业务端执行SQL语句后，要注意检查返回值，即使执行没有出错，提交的时候也可能出错。

不支持的功能特性

* 不支持存储过程、函数、触发器
* 自增id只支持在单个TIDB Server的自增，不支持多个TIDB Server的自增。
* 外键约束
* 临时表
* Mysql追踪优化器
* `XA` 语法（TiDB 内部使用两阶段提交，但并没有通过 SQL 接口公开）

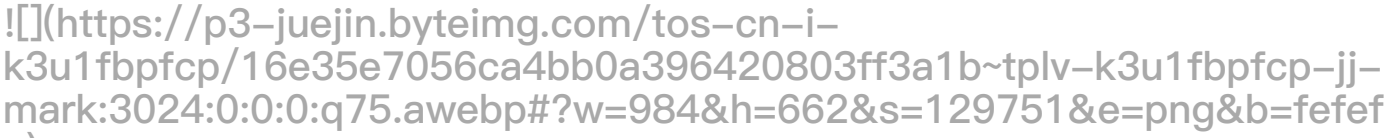
资源使用情况

以下内容参考：[pingcap.medium.com/an-8x-syste...](http://cxyroad.com/ "https://pingcap.medium.com/an-8x-system-performance-boost-why-we-migrated-from-mysql-to-a-newsql-database-a42570ab765a")

TiDB 具有很高的数据压缩比，MySQL 中的 10.8 TB 数据在 TiDB 中变成了 3.2 TB，还是三副本的总数据量。因此，**MySQL 与 TiDB 的空间使用比例为 3.4:1。**

同等量级，使用2 年以后，资源使用情况

* MySQL使用32 个节点，而 TiDB 只有 14 个
* MySql 用了 512 个 CPU 核心，而 TiDB 将仅使用 224 个，不到 MySQL 的一半。
* MySQL 使用 48 TB 存储空间，而 TiDB 将使用 16 TB，仅为 MySQL 的 1/3。



性能测试

测试报告 1

来源：[www.percona.com/blog/a-quic...](http://cxyroad.com/”https://www.percona.com/blog/a-quick-look-into-tidb-performance-on-a-single-server/”)

五个 ecs 实例，使用了不同配置，以此测试

* t2.medium：2 个 CPU 核心
* x1e.xlarge：4 个 CPU 核心
* r4.4xlarge：16 个 CPU 核心
* m4.16xlarge：64 个 CPU 核心
* m5.24xlarge：96 个 CPU 核心

MySQL 中的数据库大小为 70Gb，TiDB 中的数据库大小为 30Gb（压缩）。该表没有二级索引（主键除外）。

测试用例

1. 简单计数(*):

```
 select count(*) from ontime;
```

2. 简单分组依据

```
 select count(*), year from ontime group by year order by year;
```

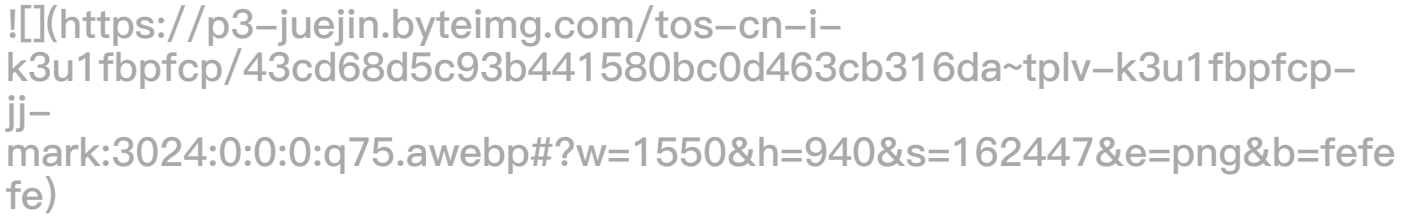
3. 用于全表扫描的复杂过滤器

```
select * from ontime where UniqueCarrier = ’DL’ and TailNum = ’N317NB’ and FlightNum = ’2’ and Origin = ’JFK’ and Dest = ’FLL’ limit 10;
```

4. 复杂的分组依据和排序依据查询

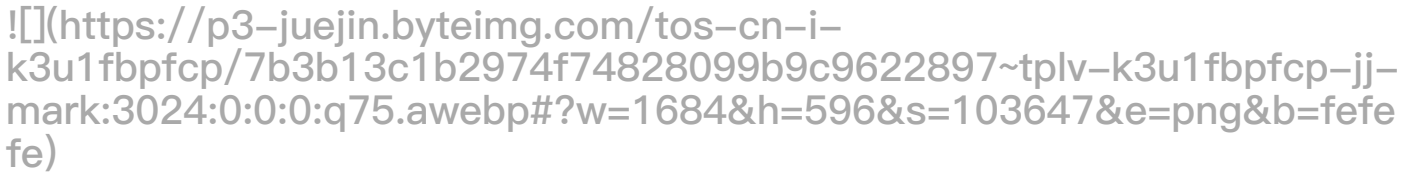
```
select SQL_CALC_FOUND_ROWS
FlightDate, UniqueCarrier as carrier,
FlightNum,
Origin,
Dest
FROM ontime
WHERE
DestState not in ('AK', 'HI', 'PR', 'VI')
and OriginState not in ('AK', 'HI', 'PR', 'VI')
and flightdate > '2015-01-01'
and ArrDelay < 15
and cancelled = 0 and Diverted = 0
and DivAirportLandings = '0'
ORDER by DepDelay DESC
LIMIT 10;
```

下图表示结果（条形表示查询响应时间，越小越好）：



系统基准测试

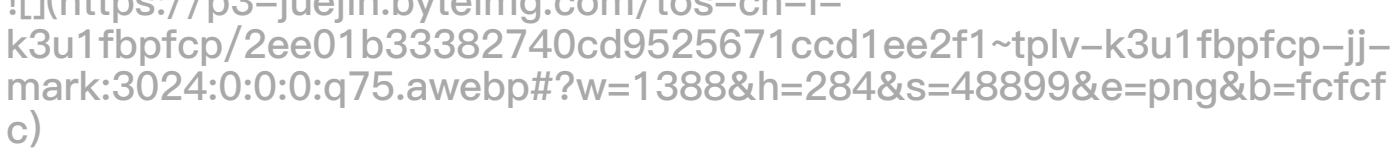
在 m4.16xlarge 实例上使用 Sysbench 进行点选择（意味着通过主键选择一行，线程范围从 1 到 128）（内存限制：无磁盘读取）。结果在这里。条形代表每秒的交易数量，越多越好：

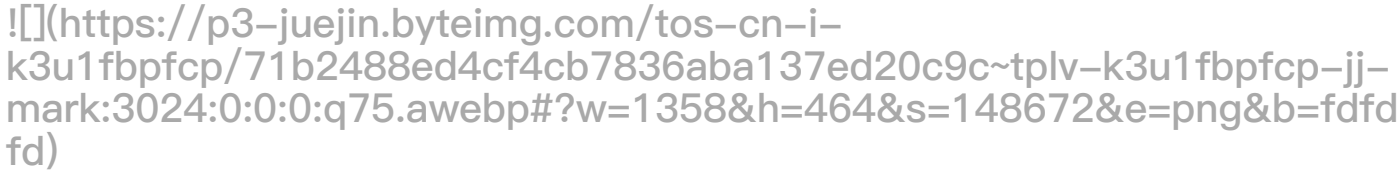


系统测试报告 2

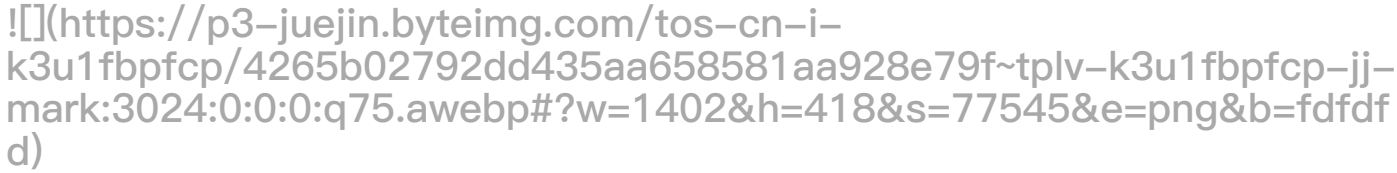
来源：[www.dcits.com/show-269-41...](http://cxyroad.com/”https://www.dcits.com/show-269-4103-1.html”)

硬件配置

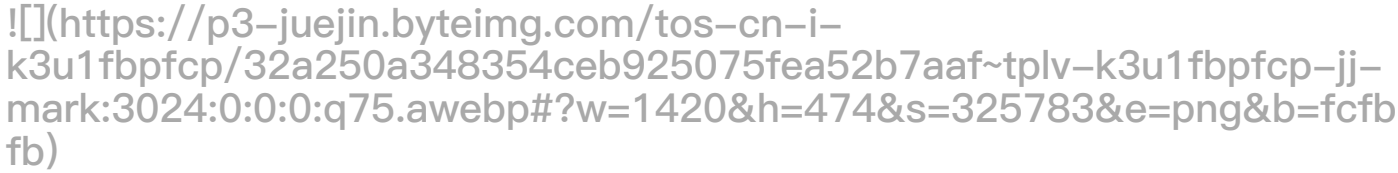


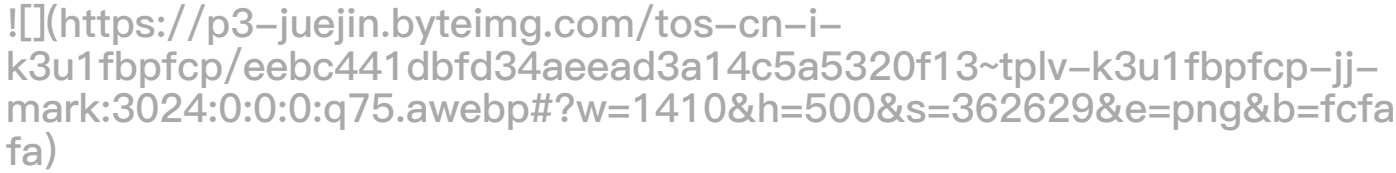


测试场景



测试分两阶段进行，第一阶段测试数据为100万单，第二阶段测试数据为1300万单。在此基础上，使用Jmeter压力测试10万单结果如下：





从测试结果来看，在小数据量mysql性能是好于TiDB，因为 TiDB 是分布式架构，如果小数据量，在网络通讯节点分发一致性等方面花的时间就很多，然后

各个节点执行完还要汇总返回，所以开销是比较大的，但是数据量一上来TiDB优势就体现出来了，所以如果数据量比较小，没必要使用 TiDB

总结

以上介绍了 TiDB架构，以及它的一些特性，同时也与 mysql 进行了对比，如果贵司的数据量比较大，正在考虑要分库分表，那么完全可以使用它，来避免分库分表，分库分表是一个过渡方案，使用分布式数据库才是终极方案。同时如果贵司的数据量比较小，那么就没必要引入了

原文链接: https://juejin.cn/post/7329413905028579340