零声教育出品 Mark 老师 QQ : 2548898954

TiDB 简介

> TiDB 是 PingCAP 公司自主设计、研发的开源分布式关系型数据库，是一款同时支持在线事务处理与在线分析处理 (Hybrid Transactional and Analytical Processing, HTAP) 的融合型分布式数据库产品，具备水平扩容或者缩容、金融级高可用、实时 HTAP、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。目标是为用户提供一站式 OLTP (Online Transactional Processing)、OLAP (Online Analytical Processing)、HTAP 解决方案。TiDB 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。

分布式系统

> 分布式系统是一种其组件位于不同的联网计算机上的系统，然后通过互相传递消息来进行通讯和协调，为了达到共同的目标，这些组件会相互作用；换句话说，分布式系统把需要进行大量计算的工程数据分割成若干个小块，由多台计算机分别进行计算和存储，然后将结果统一合并到数据结论的科学；本质上就是**进行**数据存储与计算的**分治**；

CAP 理论

一致性

> 指所有的节点在同一时间的数据一致性；
>
> all nodes see the same data at the same time;

可用性

> 服务在正常响应时间内的可用；
>
> reads and writes always succeed;

分区容错性

> 分布式系统在遇到某节点或网络分区故障的时候仍然能够对外提供满足一致性或可用性的服务；

关系型模型

> 在传统的在线交易场景里，关系型模型仍然是标准；关系型数据库的关键在于一定要具备事务；

事务

> 事务的本质是：并发控制的单元，是用户定义的一个操作序列；这些操作要么都做，要么都不做，是一个不可分割的工作单位；
>
> 为了保证系统始终处于一个完整且正确的状态；

ACID 特性

- 原子性

  > 事务包含的全部操作时一个不可分割的整体；要么全部执行，要么全部不执行；

- 一致性

> 事务的前后，所有的数据都保持一个一致的状态；不能违反数据的一致性检测；

- 隔离性

> 各个并发事务之间互相影响的程度；主要规定多个并发事务访问同一个数据资源，各个并发事务对该数据资源访问的行为；不同的隔离性是应对不同的现象（脏读、可重复读、幻读等）；

- 持久性

> 事务一旦完成要将数据所做的变更记录下来；包括数据存储和多副本的网络备份；

TiDB 部署本地测试集群

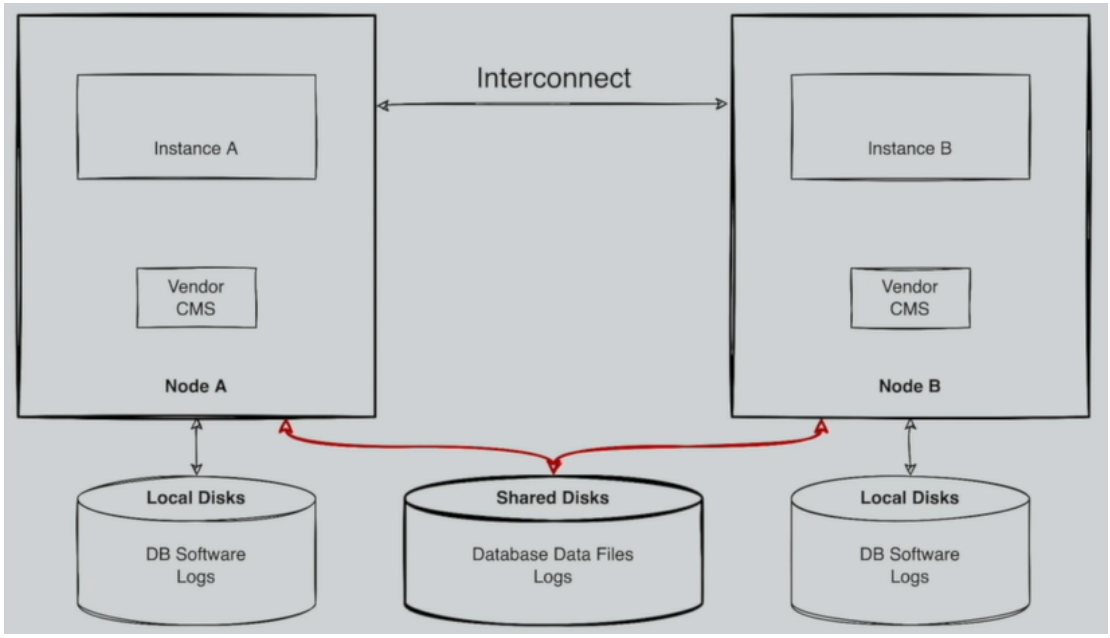
```
# 下载并安装 TiUP
curl --proto '=https' --tlsv1.2 -sSf https://tiup-mirrors.pingcap.com/install.sh | sh
# 声明全局环境变量
source .bash_profile
# 运行最新版本的 TiDB 集群，其中 TiDB、TiKV、PD 和 TiFlash 实例各 1 个
tiup playground
# 如果想指定版本并运行多个
tiup playground v4.0.16 --db 3 --pd 3 --kv 3 --monitor
# 注意：按照上面的部署，在结束部署测试后 TiUP 会清理掉原集群数据，重新执行该命令后会得到一个全新的集群。

# 如果希望持久化数据，并指定存储目录为 /tmp/tidb
tiup playground -T /tmp/tidb
```

与传统非分布式数据库架构对比

- 两者都支持 ACID、事务强一致性；
- 分布式架构，组件解耦，拥有良好的扩展性，支持弹性的扩缩容；
- 默认支持高可用，在少数副本失效的情况下，数据库能够自动进行故障转移，对业务透明；
- 采用水平扩展，在大数据量、高吞吐的业务场景中具有先天优势；
- 强项不在于轻量的简单 SQL 的响应速度，而在于大量高并发 SQL 的吞吐；

非分布式关系型数据库横向扩展案例

- 某商用数据库



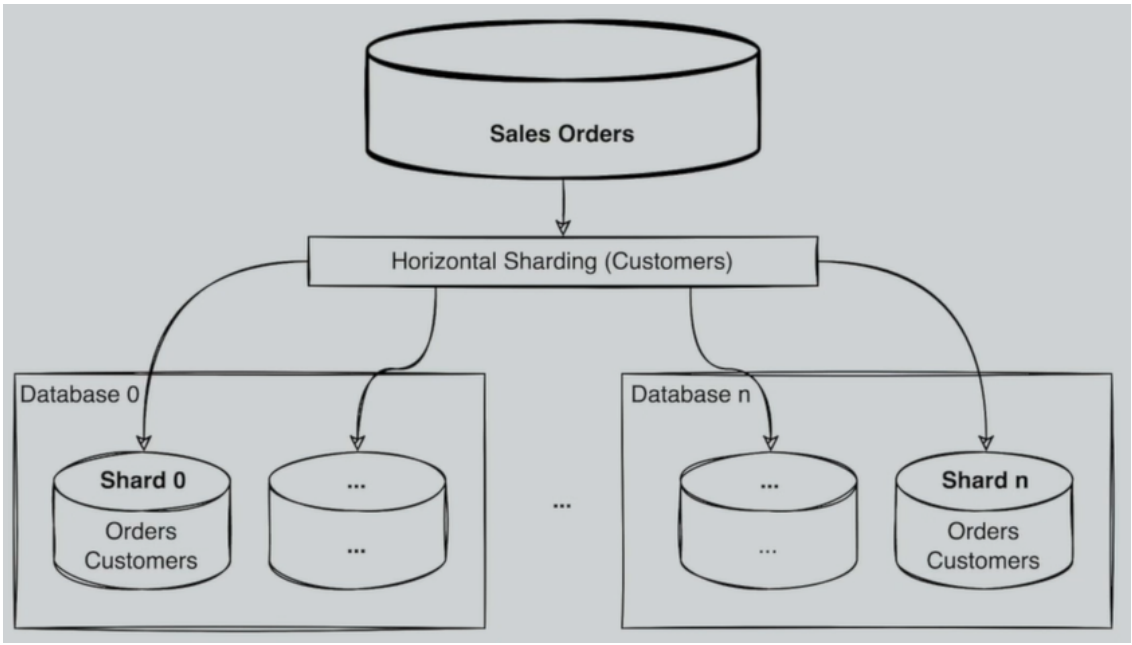
优势：

- 易于实现事务一致性；
- 无需多层复杂管理；

劣势：

- DB 节点扩展能力受限；
- 存储扩展能力及 IO 性能依赖高端共享存储；

- 水平切割



优势：

- DB 按业务类型得到水平扩展，数据得到分流

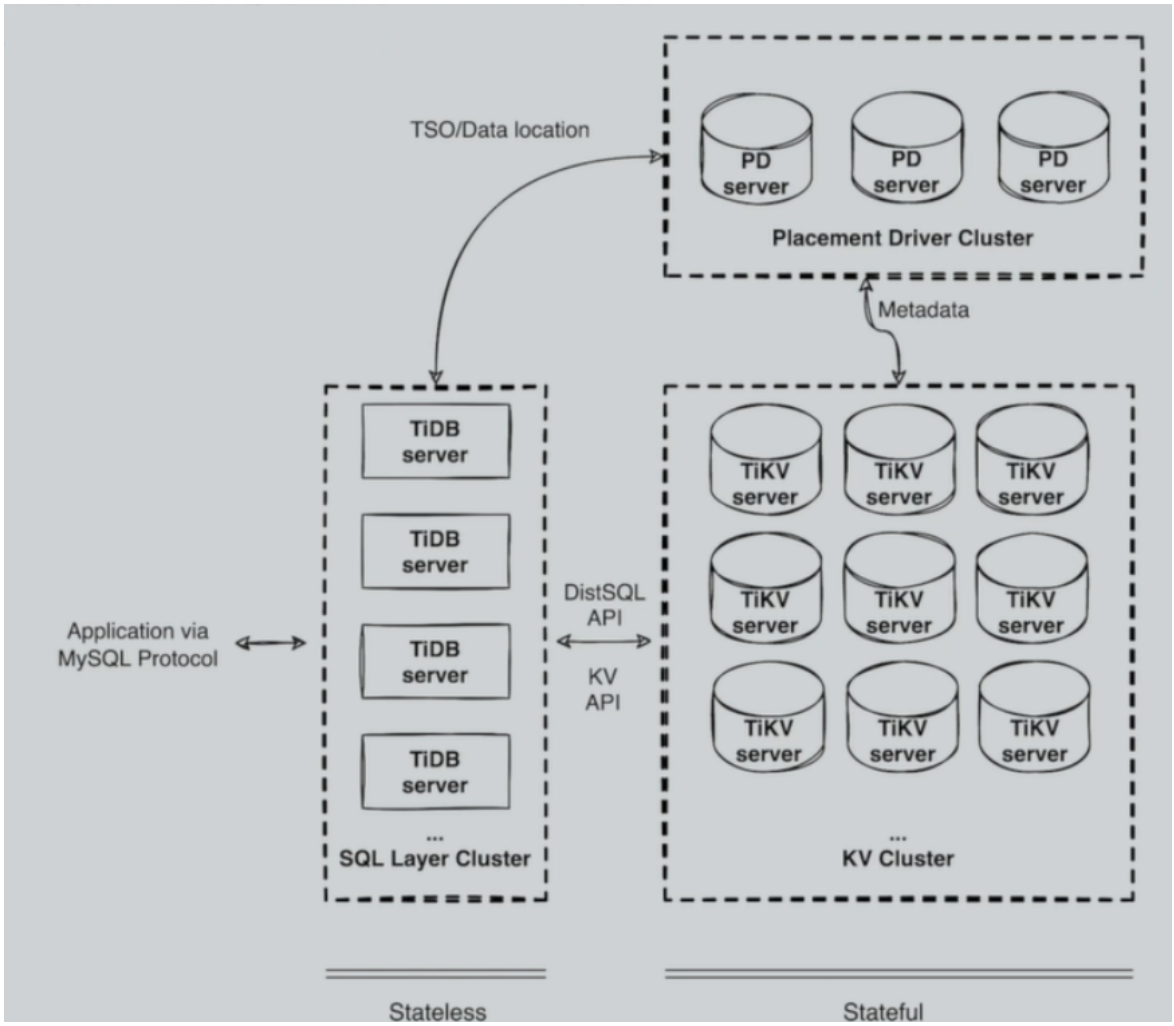
劣势：

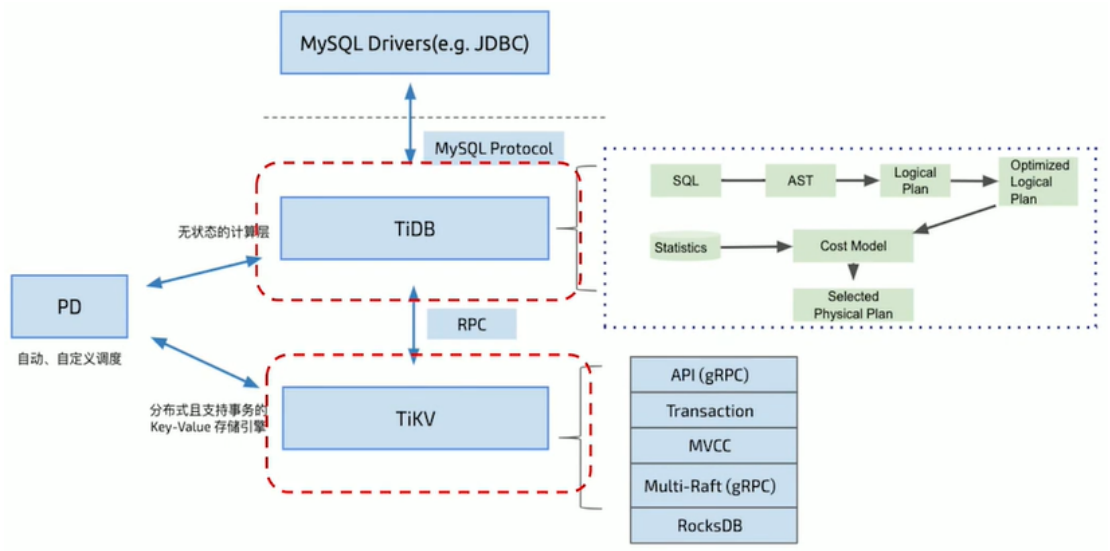
- 一定程度上退化成 NoSQL；
- 应用层需要做数据汇聚；

TiDB 分布式数据库整体架构

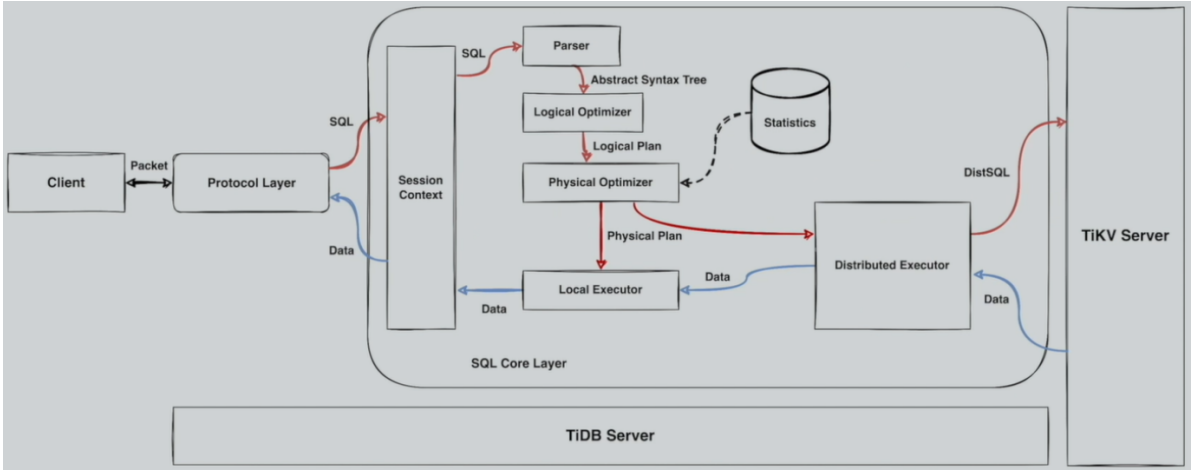
- 由多模块组成，各模块互相通信，组成完整的 TiDB 系统；
- 前端 `stateless`、后端 `stateful (Raft)`；

- 兼容 MySQL；





TiDB Server 的模块



> SQL 层，对外暴露 MySQL 协议的连接 endpoint，负责接受客户端的连接，执行 SQL 解析和优化，最终生成分布式执行计划。TiDB 层本身是无状态的，实践中可以启动多个 TiDB 实例，通过负载均衡组件（如 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。TiDB Server 本身并不存储数据，只是解析 SQL，将实际的数据读取请求转发给底层的存储节点 TiKV（或 TiFlash）。

数据映射关系

> 数据与 KV 的映射关系中定义如下：

```
tablePrefix     = []byte{'t'}
recordPrefixSep = []byte{'r'}
indexPrefixSep  = []byte{'i'}
```

> 假设表结构如下：

```
CREATE TABLE User (
    ID int,
    Name varchar(20),
    Role varchar(20),
    Age int,
    UID int,
    PRIMARY KEY (ID),
    KEY idxAge (Age),
    UNIQUE KEY idxUID (UID)
);
```

> 假设表数据如下：

```
1, "TiDB", "SQL Layer", 10, 10001
2, "TiKV", "KV Engine", 20, 10002
3, "PD", "Manager", 30, 10003
```

表数据与 KV 的映射关系

- Key 的形式：`tablePrefix{TableID}_recordPrefixSep{RowID}`；
- Value 的形式：`[col1, col2, col3, col4]`
- 映射示例：

```
t10_r1 --> ["TiDB", "SQL Layer", 10, 10001]
t10_r2 --> ["TiKV", "KV Engine", 20, 10002]
t10_r3 --> ["PD",   "Manager",   30, 10003]
```

索引数据和 KV 的映射关系

对于唯一索引：

- Key 的形式：`tablePrefix{tableID}_indexPrefixSep{indexID}_indexedColumnsValue`
- Value 的形式：`RowID`
- 映射示例：

```
t10_i1_10001 --> 1
t10_i2_10002 --> 2
t10_i3_10003 --> 3
```

非唯一索引：

- Key 的形式：`tablePrefix{TableID}_indexPrefixSep{IndexID}_indexedColumnsValue_{RowID}`
- Value 的形式：`null`
- 映射示例：

```
t10_i1_10_1 --> null
t10_i1_20_2 --> null
t10_i1_30_3 --> null
```

PD（Placement Driver）Server

> 整个 TiDB 集群的元信息管理模块，负责存储每个 TiKV 节点实时的数据分布情况和集群的整体拓扑结构，提供 TiDB Dashboard 管控界面，并为分布式事务分配事务 ID。PD 不仅存储元信息，同时还会根据 TiKV 节点实时上报的数据分布状态，下发数据调度命令给具体的 TiKV 节点，可以说是整个集群的“大脑”。此外，PD 本身也是由至少 3 个节点构成，拥有高可用的能力。建议部署奇数个 PD 节点。

调度需求

1. 作为一个分布式高可用存储系统，必须满足的需求，包括几种：

- 副本数量不能多也不能少
- 副本需要根据拓扑结构分布在不同属性的机器上
- 节点宕机或异常能够自动合理快速地进行容灾

2. 作为一个良好的分布式系统，需要考虑的地方包括：

- 维持整个集群的 Leader 分布均匀
- 维持每个节点的储存容量均匀
- 维持访问热点分布均匀
- 控制负载均衡的速度，避免影响在线服务
- 管理节点状态，包括手动上线/下线节点

> 满足第一类需求后，整个系统将具备强大的容灾功能。满足第二类需求后，可以使得系统整体的资源利用率更高且合理，具备良好的扩展性。

调度操作

- 增加一个副本
- 删除一个副本
- 将 Leader 角色在一个 Raft Group 的不同副本之间 transfer（迁移）。

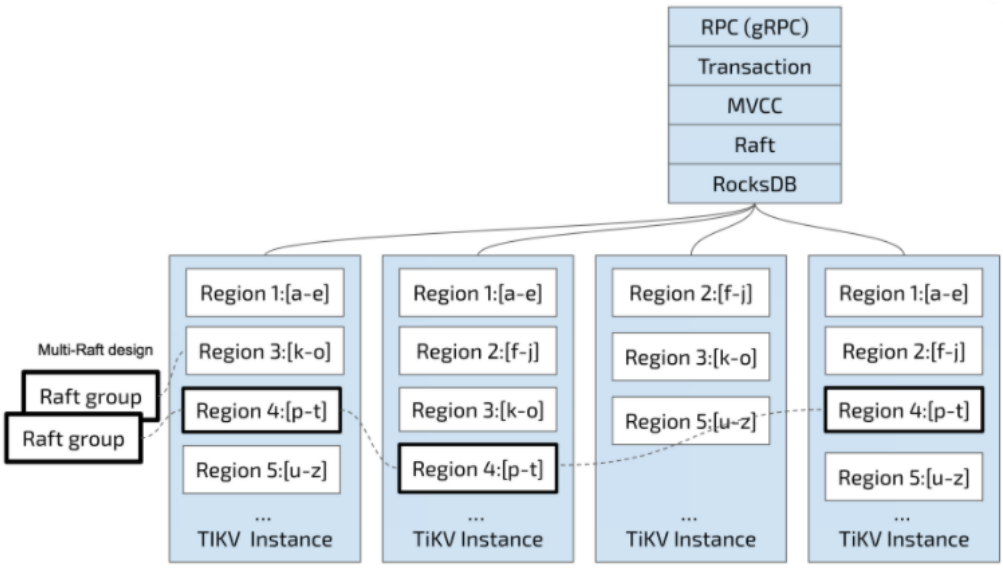
信息收集

- 每个 TiKV 节点会定期向 PD 汇报节点的状态信息
- 每个 Raft Group 的 Leader 会定期向 PD 汇报 Region 的状态信息

存储节点

TiKV Server

> 负责存储数据，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 的 API 在 KV 键值对层面提供对分布式事务的原生支持，默认提供了 SI (Snapshot Isolation) 的隔离级别，这也是 TiDB 在 SQL 层面支持分布式事务的核心。TiDB 的 SQL 层做完 SQL 解析后，会将 SQL 的执行计划转换为对 TiKV API 的实际调用。所以，数据都存储在 TiKV 中。另外，TiKV 中的数据都会自动维护多副本（默认为三副本），天然支持高可用和自动故障转移。



TiFlash

> TiFlash 是一类特殊的存储节点。和普通 TiKV 节点不一样的是，在 TiFlash 内部，数据是以列式的形式进行存储，主要的功能是为分析型的场景加速。