TenDB Cluster分部署数据库

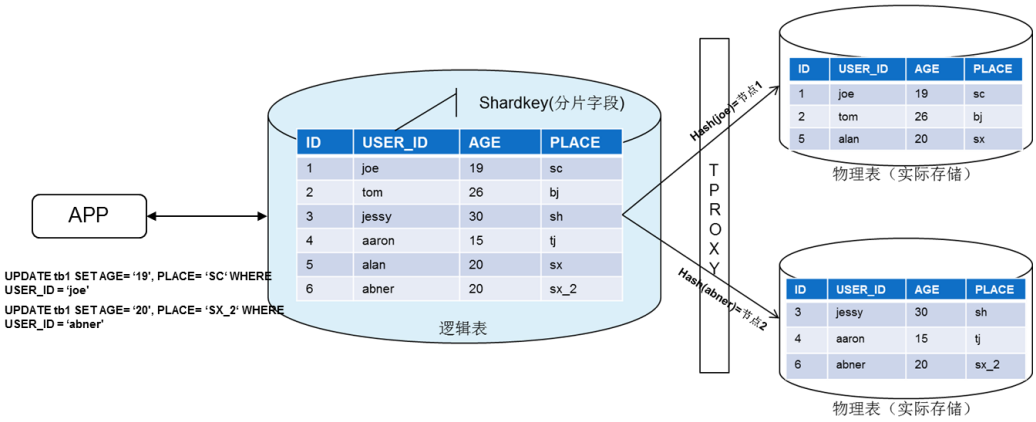
https://tendbcluster.com/book-cn/

随着网站的壮大，MySQL数据库架构一般会经历如下图所示的演进过程。

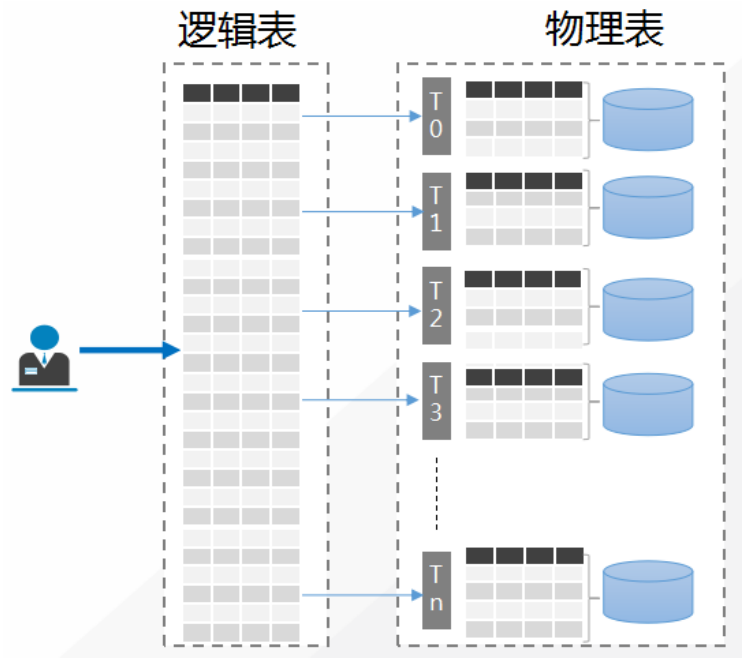
分库分表的技术解决方案总体上分为两大类：应用层依赖类中间件和中间层代理类中间件。

1）应用层依赖类中间件
通过应用层修改代码来实现分库分表，它使用客户端直连数据库，以jar包形式提供服务，无需额外部署和依赖，可理解为增强版的JDBC驱动。
优点：在高并发请求下，性能有一定的优势，可以减少一层网络交互。
缺点：不能跨语言，比如Java写的ShardingSphere-JDBC显然不能运用在PHP/Golang项目中。

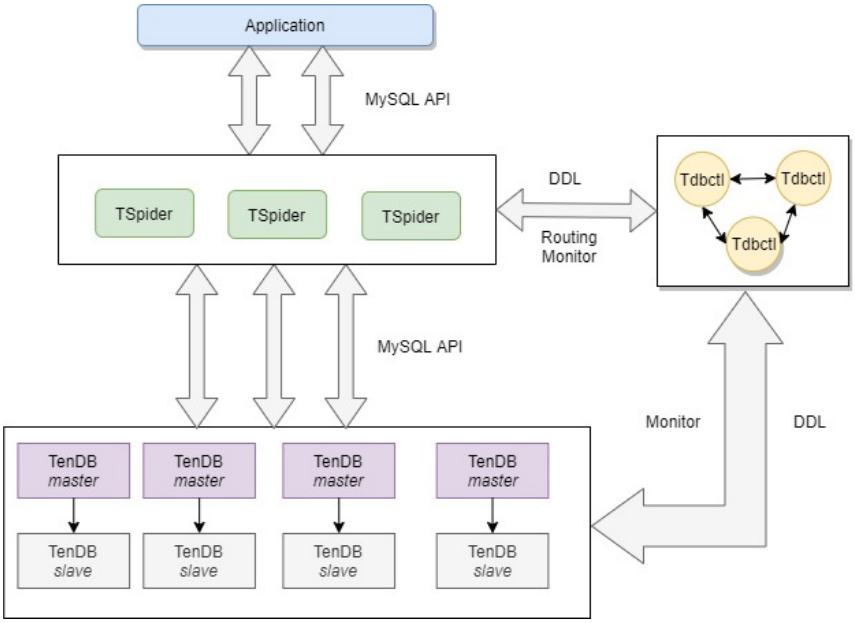
2）中间层代理类中间件
位于前端应用和数据库之间，前端应用以标准的MySQL协议来连接代理层，并按照数据分片规则转发请求到后端MySQL数据库中，再合并结果集返回应用端， 对应用程序完全透明，无需修改业务代码，就像访问单台MySQL数据库一样。
缺点：在高并发请求下，性能损耗略高于应用层依赖类中间件，多了一层网络交互。
优点：应用层语言无限制，支持在Java和PHP/Golang项目中运行。



水平切分（分库分表）：是按照某种规则，将一个表的数据分散到多个物理独立的数据库服务器中，形成“独立”的数据库“分片”。 多个分片共同组成一个逻辑完整的数据库实例。 分布式数据库中，根据在建表时设定的分表键，系统将根据不同分表键自动分布数据到不同的物理分片中，但逻辑上仍然是一张完整的表。



TenDB Cluster整体架构概述



注：不支持自动迁移数据，通常是在初期就把分片确定好，规划好3年的数据增长量，制定分片规则，不建议变更分片数。例如对字段uid进行取模，假设原来的值为64，现在变更为256，如果要变更分片规则，就需要停机重导数据。

TenDB Cluster工作原理

作为一种MySQL引擎，TSpider天然地支持MySQL协议，使用MySQL标准API即可请求TSpider。
TSpider在收到应用请求后，会通过数据路由规则对SQL语句进行改写，然后分发到相应的存储节点MySQL/MariaDB中执行，再对MySQL或MariaDB的返回结果进行处理，并最终返回给应用层。

TSpider本身并不存储数据，基本上是无状态的（各TSpider节点的配置应有所不同），可无限水平扩展。应用层可通过负载均衡组件（比如LVS、Haproxy、DNS轮询）提供的统一接入地址访问多个对等的TSpider节点。

应用程序连接TSpider时，TSpider充当中间件代理，将客户端查询的请求按照事先定义好的分片规则分发给后端数据库，之后返回的数据会在TSpider内存里汇总，并最终返回给客户端请求，这个过程对于应用程序而言是透明的。

1）代理入口层：**TSpider**是TenDB Cluster的接入层，是腾讯游戏基于MariaDB 10.3.7开发定制的版本，主要完善并定制spider这一分布式MySQL存储引擎；

https://github.com/Tencent/TenDBCluster-TSpider/releases

2）表结构修改层：**Tdbctl**是集群的中央控制模块，基于MySQL 5.7开发，业务在TSpider中执行的DDL操作会由TSpider转发到Tdbctl，在Tdbctl中进行SQL改写后会分别分发到TSpider**（将用户创建的表自动转为Spider引擎）**和后端MySQL/MariaDB的各节点**（将用户创建的表自动转为InnoDB引擎，并按照哈希规则创建表）**上执行。
Tdbctl支持MySQL的MGR特性，因此在部署上可以使用3个Tdbctl节点，搭建成一个MGR集群，也可以用两个节点搭建主从复制架构，从而保证中控节点Tdbctl的高可用性及路由配置的一致性。
https://github.com/Tencent/TenDBCluster-Tdbctl/releases/

3）数据存储层：MySQL/Percona 5.7、8.0/MariaDB 10.3+

TenDB Cluster集群部署

1）TSpider安装

```
# 下载腾讯版MariaDB TSpider
```

```
Shell> wget
https://github.com/Tencent/TenDBCluster-
TSpider/releases/download/tspider-3.7.4/mariadb-10.3.7-
linux-x86_64-tspider-3.7.4-gcs.tar.gz

# 解压软链介质
Shell> tar xzvf mariadb-10.3.7-linux-x86_64-tspider-3.7.4-
gcs.tar.gz -C /usr/local/
Shell> ln -s mariadb-10.3.7-linux-x86_64-tspider-3.7.4-gcs
tspider
Shell> chown -R mysql.mysql mariadb-10.3.7-linux-x86_64-
tspider-3.7.4-gcs/

# 初始化TSpider
Shell> cd /usr/local/tspider && ./scripts/mysql_install_db --
defaults-file=/etc/my_tspider.cnf
--user=mysql

# 启动TSpider
Shell> ./bin/mysqld_safe --defaults-file=/etc/my_tspider.cnf -
-user=mysql &
```

TSpider引擎的重要参数说明如下

> □spider_conn_recycle_mode= 1

连接复用，类似于连接池的功能。

> □optimizer_switch= 'engine_condition_pushdown=on'

引擎下推，将查询推送到后端数据库，然后将查询结果返回给TSpider做聚合。

> □spider_max_connections

用于控制TSpider节点连接后端MySQL/MariaDB服务器的最大连接数。该参数默认取值为0，表示对最大连接数没有限制。

> □spider_bgs_mode

TSpider集群在接受应用层的SQL语句请求后，将判定SQL语句需要路由到后端的哪些MySQL/MariaDB实例上执行，然后在对应的后端MySQL/MariaDB实例上依次轮询执行，最后统一汇总结果。该参数取值为0时表示不开启并行功能（即串行执行）；取值为1时表示开启并行功能。

> □spider_bgs_dml

当spider_bgs_mode取值为1，即开启并行功能时，spider_bgs_dml的值设置为1表示插入（insert）、修改（update），和删除（delete）操作也开启并行功能；否则不开启并行功能。

> □spider_auto_increment_mode_switch

用于设置是否启用由TSpider控制主键自增键值（只保证自增ID的唯一性，不保证ID的连续性和递增性）。该参数的默认值为1，在生产环境中不用修改。

> □spider_auto_increment_mode_value

用于设置主键自增的起始值，集群各TSpider节点的配置不能重复。

> spider_auto_increment_step

用于设置主键自增的起始步长，集群各TSpider节点的配置相同。
例如，第一个TSpider节点参数如下：
spider_auto_increment_mode_switch=1
spider_auto_increment_mode_value=1
spider_auto_increment_step=17
那么，此TSpider节点上的主键自增键值依次为1、18、35、52……

第二个TSpider节点参数如下：
spider_auto_increment_mode_switch=1
spider_auto_increment_mode_value=2
spider_auto_increment_step=17
那么，此TSpider节点上的主键自增键值依次为 2、19、36、53 ……

> ddl_execute_by_ctl

当将该参数的值设置为ON的时候，数据库管理员在TSpider节点上执行的DDL语句会路由给Tdbctl，由Tdbctl对集群中的TSpider和后端MySQL/MariaDB节点进行统一变更处理。当将该参数的值设置为OFF的时候，中控节点Tdbctl不会对DDL进行转发，需要分别在TSpider和后端MySQL/MariaDB上执行DDL操作。

2）Tdbctl安装

```
# 下载Tdbctl
Shell> wget
https://github.com/Tencent/TenDBCluster-
Tdbctl/releases/download/tdbctl-2.1/mysql-5.7.20-linux-
x86_64-tdbctl-2.1.tar.gz

# 解压软链介质
Shell> tar xzvf mysql-5.7.20-linux-x86_64-tdbctl-2.1.tar.gz -C
/usr/local/
Shell> ln -s mysql-5.7.20-linux-x86_64-tdbctl-2.1 tdbctl
Shell> chown -R mysql.mysql mysql-5.7.20-linux-x86_64-
tdbctl-2.1/

# 初始化Tdbctl
Shell> cd /usr/local/ tdbctl && ./bin/mysqld --defaults-
file=/etc/my_tdbctl.cnf
--initialize --user=mysql

# 启动Tdbctl
Shell> ./bin/mysqld_safe --defaults-file=/etc/my_tdbctl.cnf --
user=mysql &
```

Tdbctl重要参数说明如下

> tc_is_primary

该参数取值为1时表示当前节点为主节点，允许执行集群相关的DDL语句、管理语句等；如果Tdbctl节点配置了MGR复制模式，此值就会基于MGR的算法自动设置为1；否则就要显式设定此参数的值为1，以指定当前节点为主节点。

> tc_mysql_wrapper_prefix

中控节点在构建TSpider节点上执行的建表语句时，需要按照一定的分片顺序来进行。该参数可用于约束集群中存储节点的路由信息对应的Server_name的前缀，如果前缀为backend，则存储节点Server_name的写法必须是backend0、backend1、backend *n*等，数字部分为从0开始的连续整数。

```
tc_check_repair_routing
```

该参数取值为“ON”时表示中控节点会自动检查TSpider节点中的路由配置和中控节点是否一致，如果不一致就修正到一致。

Tdbctl配置管理

1）连接Tdbctl节点，配置mysql.servers路由表，命令如下：

```
Shell> mysql -umysql -pmysql -h192.168.0.9 -P26000  mysql -A
```

2）插入路由信息，命令如下：

```
> -- 定义后端MySQL/MariaDB节点
insert into mysql.servers
values('backend0','192.168.71.12','','mysql','mysql',3306,'','mysql','');
insert into mysql.servers
values('backend1','192.168.71.13','','mysql','mysql',3306,'','mysql','');

> -- 定义TSpider节点
insert into mysql.servers
values('SPIDER0','192.168.71.11','','mysql','mysql',25000,'','SPIDER','');

> -- 定义Tdbctl节点
insert into mysql.servers
values('TDBCTL0','192.168.71.11','','mysql','mysql',26000,'','TDBCTL','');
```

注意，这里需要在各个节点上分别创建用户名“mysql”，密码也为“mysql”。

3）刷新路由，将路由同步到TSpider节点，命令如下：

```
> tdbctl flush routing;
```

注意，如果这一步发生报错，请查询error.log错误日志来排查问题。

TSpider引擎在分库分表中的应用

创建user表并指定uid字段做取模分片，命令如下：

```
CREATE TABLE user (
  id int(11) unsigned NOT NULL AUTO_INCREMENT,
  uid int(11) NOT NULL,
  PRIMARY KEY (id,uid),
  KEY uid (uid)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
COMMENT='shard_key "uid"';
```

当取模结果为0时，数据将写入后端backend1服务器上user_0库下的user1表里。

当取模结果为1时，数据将写入后端backend2服务器上user_1库下的user2表里。

分片键的选取建议具体如下。

> 1、分片的字段最好是业务经常查询的条件字段，这样可以提高查询效率。
> 2、数据应该均匀分布，避免所有数据集中在一个分片上。