分布式数据库 TDSQL

最佳实践

产品文档

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

文档目录

最佳实践
从单机实例导入到分布式实例

最近更新时间：2020-12-23 19:01:16

由于分布式数据库的分布式架构对用户透明，一般情况下，只需要预先建好表结构。可以使用 mysqldump、或其他 Navicat、SQLyog 等 MySQL 客户端进行迁移。迁移步骤如下：

第一步：准备导入导出环境
第二步：导出源表的表结构和数据
第三步：修改建表语句并在目的表中创建表结构
第四步：导入数据

步骤1：准备导入导出环境

准备迁移数据前，您需要准备好如下环境：

- 云服务器
  - 建议配置 CPU 2核，内存8GB，磁盘500GB以上（取决于数据量大小）
  - Linux
  - 安装 MySQL 客户端
  - 如果您迁移的数据量较小（< 10GB），也可以通过外网（互联网）直接导入，无需准备
- TDSQL MySQL版
  - 根据预期选择大小，并根据源库字符集，表名大小写，innodb_page_size 大小进行初始化
  - 创建帐号，该帐号建议开启全局所有权限
  - 必要时开启外网 IP

步骤2：导出源表的表结构和数据

演示环境

- 操作库：caccts
- 操作表：t_acct_water_0
- 源库：单实例 MySQL
- 目标库：TDSQL for Percona、MariaDB

在源库导出表结构

通过命令 `mysqldump -u username -p password -d dbname tablename > tablename.sql` 导出表结构。

```
//命令实例
mysqldump -utest -ptest1234 -d -S /data/4003/prod/mysql.scok caccts t_acct_water_0 > table.sql
```

在源库导出表数据

通过命令 `mysqldump -c -u username -p password dbname tablename > tablename.sql` 导出表数据。

```
//命令实例
mysqldump -c -t -utest -ptest1234 -S /data/4003/prod/mysql.scok caccts t_acct_water_0 > data.sql
```

> ⚠ 注意：
> 导出数据必须通过 mysqldump 工具导出，并且加上 -c 参数，因为这样导出的数据行都带有列名字段，不带列名字段的 sql 会被 TDSQL for Percona、MariaDB 拒绝掉。-t 参数的意义是不导出表结构，只导出表数据。

上传文件至云服务器某目录

上传前，您需开启 CVM 外网访问地址，并参见 Linux 系统通过 SCP 上传文件到 Linux 云服务器 上传文件，您至少需要上传刚刚导出的：

- 表结构 sql：table.sql
- 数据 sql：data.sql

步骤3：修改建表语句并在目的表中创建表结构

打开刚导出的表结构文件 table.sql，参考如下格式语句添加主键和 shardkey，并另存为 tablenew.sql。

```
CREATE TABLE (
列名称1 数据类型,
列名称2 数据类型,
列名称3 数据类型,
....,
PRIMARY KEY('列名称n'))
ENGINE=INNODB DEFAULT CHARSET=xxxx
shardkey=keyname
```

> **注意：**
> 必须要设置主键，必须指定 shardkey，必须注意表名大小问题，建议删除多余注释，否则建表可能不成功。

```
CREATE TABLE `t_acct_water_0` (
  `Fseq_no` int(11) NOT NULL DEFAULT '0',
  `Facct_id` varchar(64) NOT NULL DEFAULT '',
  `Facct_key` varchar(128) NOT NULL DEFAULT '',
  `Ffirstkey` varchar(64) NOT NULL DEFAULT '',
  `Fuin` varchar(64) NOT NULL DEFAULT '',
  `Fuserid` varchar(64) NOT NULL DEFAULT '',
  `Factionid` varchar(32) NOT NULL DEFAULT '',
  `Fzoneid` varchar(32) NOT NULL DEFAULT '',
  `Froleid` varchar(32) NOT NULL DEFAULT '',
  `Fsubacctid` varchar(64) NOT NULL DEFAULT '',
  `Fextern_tran_type` int(11) NOT NULL DEFAULT '0',
  `Fbase_tran_type` int(11) NOT NULL DEFAULT '0',
  `Fwater_type` int(11) NOT NULL DEFAULT '0',
  `Fio_flag` int(11) NOT NULL DEFAULT '0',
  `Fbill_no` varchar(64) NOT NULL DEFAULT '',
  `Fportal_seq` varchar(256) NOT NULL DEFAULT '',
  `Ftran_amt` bigint(20) NOT NULL DEFAULT '0',
  `Fbalance` bigint(20) NOT NULL DEFAULT '0',
  `Fgen_tran_amt` bigint(20) NOT NULL DEFAULT '0',
  `Fgen_balance` bigint(20) NOT NULL DEFAULT '0',
  `Fcreate_time` int(11) NOT NULL DEFAULT '0',
  `Fsource` varchar(32) NOT NULL DEFAULT '',
  `Fdevice` varchar(32) NOT NULL DEFAULT '',
  `Fcheck_account` varchar(256) NOT NULL DEFAULT '',
  `Fclient_ip` varchar(32) NOT NULL DEFAULT '',
  `Fuser_ip` varchar(32) NOT NULL DEFAULT '',
  `Ftran_info` varchar(256) NOT NULL DEFAULT '',
  `Fremark` varchar(256) NOT NULL DEFAULT '',
  `Freserve1` varchar(256) NOT NULL DEFAULT '',
  `Freserve2` varchar(256) NOT NULL DEFAULT
  `blobtest` blob,
  PRIMARY KEY (`Fseq_no`),
  KEY `Ffirstkey` (`Ffirstkey`),
  KEY `s_acctid_acctkey` (`Facct_id`,`Facct_key`),
  KEY `i_create_time` (`Fcreate_time`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 shardkey=Fseq_no
```

步骤4：导入数据

连接 TDSQL for Percona、MariaDB 实例

在 CVM 上使用 `mysql -u username -p password -h IP -P port` 登录 MySQL 服务器，然后使用 `use dbname` 进入数据库。

> ⚠ 注意：
> 您可能需要先创建库。

导入表结构

使用刚刚上传的文件，用 `source` 命令导入数据。

1. 先导入表结构：`source /文件路径/tablenew.sql;`
2. 再导入数据：`source /文件路径/data.sql;`
3. 校验导入情况：`select count(*) from tablename;`

> ⚠ 注意：
> 需先导入建表语句，再导入数据。也可以通过 mysql 的 source 命令直接导入 sql。

其他方案

整体来说，只要能够在导入数据前，在目的表先创建对应的表结构（需指定 shardkey），就可以比较顺利的导入数据。

从分布式实例导入到分布式实例

最近更新时间：2020-12-23 19:01:21

由于分布式数据库到分布式数据库的数据导入方案与一般情况不同，使用 mysqldump 对分布式实例导入数据到分布式实例的步骤如下：

1. 安装 MariaDB 版本的 mysqldump

购买 Linux 版的云服务器，使用 `yum install mariadb-server` 即可安装。

2. 导出表结构

```
mysqldump --compact --single-transaction -d -uxxx -pxxx -hxxx.xxx.xxx.xxx -Pxxxx db_name table_name > schema.sql
```

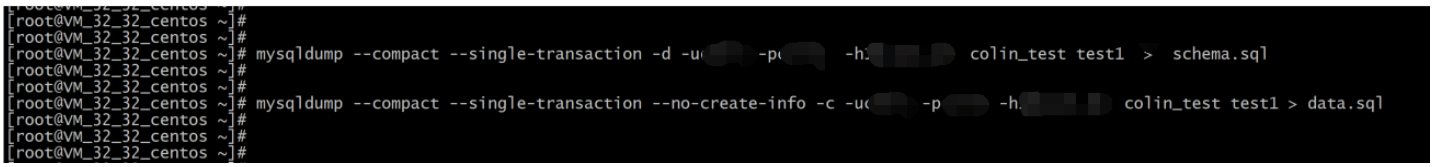
> 说明：
> db_name 和 table_name 参数根据实际情况选择。

3. 导出数据

mysqldump 导出数据：
在 TDSQL MySQL版 控制台 的参数设置中设置 net_write_timeout 参数：set global net_write_timeout=28800

```
mysqldump --compact --single-transaction --no-create-info -c -uxxx -pxxx -hxxx.xxx.xxx.xxx -Pxxxx db_name table_name > data.sql
```

> 说明：
> db_name 和 table_name 参数根据实际情况选择，如果导出的数据要导⊠到另外⊠套 TDSQL MySQL版 环境的话，必须加上 -c 选项，-c 与 db_name 之间需添加空格。



4. 在目标库创建 db

```
mysql --default-character-set=utf8 -uxxx -pxxx -hxxx.xxx.xxx.xxx -Pxxxx -e "create database dbname;";
```

- --default-character-set=utf8：根据您目标表实际情况设定。
- -uxxx：有权限的帐号（-u 是关键字）。
- -pxxx：密码（-p 是关键字）。
- -hxxx.xxx.xxx.xxx -Pxxxx：数据库实例的 IP 和端口。
- dbname：代表 db 的名字。

5. 在目标库上导入表结构

```
mysql --default-character-set=utf8 -uxxx -pxxx -hxxx.xxx.xxx.xxx -Pxxxx dbname < schema.sql
```

- --default-character-set=utf8：根据您目标表实际情况设定。
- -uxxx：有权限的帐号（-u 是关键字）。
- -pxxx：密码（-p 是关键字）。
- -hxxx.xxx.xxx.xxx -Pxxxx：数据库实例的 IP 和端口。
- dbname：代表 db 的名字。

6. 在目标库上导入表数据

```
mysql --default-character-set=utf8 -uxxx -pxxx -hxxx.xxx.xxx.xxx -Pxxxx dbname < data.sql
```

说明：

如果源表中使用了自增字段，并且导入的时候出现“Column 'xx' specified twice”的错误，则需要对 schema.sql 做处理。

去掉自增字段的反引号(cat schema.sql | tr "\`" " " > schema_tr.sql)，然后 drop database，使用处理后的 schema_tr.sql 重复步骤3 – 5的操作。

```
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]# mysql --default-character-set=utf8 -u    -p    -h     -e "create database colin_test;"
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]# mysql --default-character-set=utf8 -u    -p    -h     colin_test <  schema.sql
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]# mysql --default-character-set=utf8 -u    -p    -h     colin_test <  data.sql
[root@VM_32_32_centos ~]#
```

选择实例配置和分片配置

最近更新时间：2020-12-23 19:01:25

TDSQL MySQL版 选型概述

TDSQL MySQL版 由分片（sharding）组成，分片的规格和分片数量决定了 TDSQL MySQL版 实例的处理能力。理论上来讲：

- TDSQL MySQL版 实例读写并发性能 = ∑（某规格分片性能 * 某规格分片数量）
- TDSQL MySQL版 实例事务性能 = ∑（某规格分片事务性能 * 70% * 某规格分片数量）

因此，分片规格越高、分片数量越多，实例的处理能力越强。而分片性能，主要与 CPU / 内存 相关，并以 QPS 为基础衡量指标，我们在分片性能说明章节，给出了大致性能指标。

TDSQL MySQL版 分片规格的选择

TDSQL MySQL版 分片规格的选择，主要从三个方面需求来决定：1、性能需求；2、容量需求；3、其他要求。

性能需求：通过预判至少6个月的性能规模和可能增长，您可以确定您分布式实例所需总 CPU / 内存 规模。
容量需求：通过预判至少1年的容量规模和可能增长，您可以确定您分布式实例所需总 磁盘 规模。
其他要求：我们建议一个分片**至少存储5000W行数据**，并考虑到业务中所需的 广播表、单表，和节点内 join 等业务需求。

> ⚠ 注意：
> 建议您先确保让单个分片配置较大，而分片数量较少。

综合上述来看，我们预估您可能有如下几种业务特点，推荐策略如下：

- 使用 TDSQL MySQL版 做功能性测试，且对性能没有特别要求：2个分片，每个分片配置为：**内存/磁盘：2GB/25GB**。
- 业务发展初期，总数据规模较小但增长快的选型：2个分片，每个分片配置为：**内存/磁盘：16GB/200GB**。
- 业务发展稳定，根据业务实际情况选型：4个分片，每个分片配置等于：**当前业务峰值*增长率/4**

TDSQL MySQL版 分片性能测试

数据库基准性能测试为 sysbench 0.5 工具修改说明：对 sysbench 自带的 otlp 脚本做了修改，读写比例修改为 1：1，并通过执行测试命令参数 oltp_point_selects 和 oltp_index_updates 控制读写比例，本文测试用例的均采用4个 select 点，1个 update 点，读写比例保持 4:1。

vCPU（核）	内存(GB)	存储空间(GB)	数据集(GB)	客户端数	单客户端并发数	QPS	TPS
1	2GB	100GB	46GB	1	128	1880	351
2	4GB	200GB	76GB	1	128	3983	797
2	8GB	200GB	142GB	1	128	6151	1210
4	16GB	400GB	238GB	1	128	10098	2119
4	32GB	700GB	238GB	2	128	20125	3549
8	64GB	1T	378GB	2	128	37956	7002
12	96GB	1.5T	378GB	3	128	51026	10591
16	120GB	2T	378GB	3	128	81050	15013
24	240GB	3T	567GB	4	128	96891	17698
48	480GB	6T	567GB	6	128	140256	26599

此处 TPS 为单机 TPS，并非测试的是分布式事务的 TPS。
根据运营策略要求，当前 TDSQL MySQL版 的（部分）实例都采用闲时超用技术，所以您可能在您的监控中看到 CPU 利用率超过100%的情况。