MySQL学习笔记（Day020：InnoDB_1–表空间/General）

MySQL学习

一. 作业解析

- **在MySQL5.6的版本中找到线程号**

```
shell> gdb -ex "set pagination 0" -ex "thread apply all bt" --batch -p [ 进程ID or 线程ID ]

# -p  : 指定attach到哪个进程或者线程中
# --batch : 使用批处理模式，在所有的命令执行完成后退出，正确返回0，错误为 非0
# --ex :  执行后面跟着的GDB命令
# set paginagtion 0:  不使用分页输出
# thread apply all bt :  bt是backtrace的简写，即对所有线程使用backtrace命令，
#                        bracktrace命令会产生一张表，包含当前的函数调用等信息


# 可写成脚本的方式后，直接添加进程ID作为参数
shell> vim gdb_tid_print.sh  # <== 将刚才的命令写入到脚本文件中

shell> cat gdb_tid_print.sh
#!/bin/bash
gdb -ex "set pagination 0" -ex "thread apply all bt" --batch -p $1  # $1 b表示该脚本的第一个参数

shell> chmod +x gdb_tid_print.sh
shell> ./gdb_tid_print.sh 2495  # 2495 是我的mysql5.6.27的进程 通过ps -ef | grep mysql获得
[root@MyServer ~]# sh ./gdb_tid_print.sh 2495
# 下面这些就是所有的thread_os_id
[New LWP 2521]
[New LWP 2519]
[New LWP 2518]
[New LWP 2517]
[New LWP 2516]
[New LWP 2515]
[New LWP 2514]
[New LWP 2513]
[New LWP 2512]
[New LWP 2511]
[New LWP 2510]
[New LWP 2509]
[New LWP 2508]
[New LWP 2507]
[New LWP 2505]
[New LWP 2504]
[New LWP 2503]
[New LWP 2502]
[New LWP 2501]
[New LWP 2500]
[New LWP 2499]
[New LWP 2498]
[New LWP 2497]
[New LWP 2496]
[Thread debugging using libthread_db enabled]
0x000000328f6df343 in poll () from /lib64/libc.so.6

Thread 25 (Thread 0x7fc3d5db6700 (LWP 2496)):  ## LWP就是 LightWeight Process 的意思
#
# 输出的内容可以知道该线程在做什么，需要能看的懂代码
#
#0  0x000000328f200614 in ?? () from /lib64/libaio.so.1
#1  0x0000000000974a9f in os_aio_linux_collect (global_seg=0, message1=0x7fc3d5db5e58, message2=0x7fc3d5db5e50, type=0x7fc3d5db5e48) at /export/home/pb2/build/sb_0-16513091-1442592600.21/mysql-5.6.27/storage/innobase/os/os0file.cc:4975
#2  os_aio_linux_handle (global_seg=0, message1=0x7fc3d5db5e58, message2=0x7fc3d5db5e50, type=0x7fc3d5db5e48) at /export/home/pb2/build/sb_0-16513091-1442592600.21/mysql-5.6.27/storage/innobase/os/os0file.cc:5119
#3  0x0000000000a7732e in fil_aio_wait (segment=0) at /export/home/pb2/build/sb_0-16513091-1442592600.21/mysql-5.6.27/storage/innobase/fil/fil0fil.cc:5754
#4  0x00000000009dbbb8 in io_handler_thread (arg=<value optimized out>) at /export/home/pb2/build/sb_0-16513091-1442592600.21/mysql-5.6.27/storage/innobase/srv/srv0start.cc:489
#5  0x000000328fa079d1 in start_thread () from /lib64/libpthread.so.0
#6  0x000000328f6e8b6d in clone () from /lib64/libc.so.6

Thread 24 (Thread 0x7fc3d53b5700 (LWP 2497)):
#0  0x000000328f200614 in ?? () from /lib64/libaio.so.1
#1  0x0000000000974a9f in os_aio_linux_collect (global_seg=1, message1=0x7fc3d53b4e58, message2=0x7fc3d53b4e50, type=0x7fc3d53b4e48) at /export/home/pb2/build/sb_0-16513091-1442592600.21/mysql-5.6.27/storage/innobase/os/os0file.cc:4975
#2  os_aio_linux_handle (global_seg=1, message1=0x7fc3d53b4e58, message2=0x7fc3d53b4e50, type=0x7fc3d53b4e48) at /export/home/pb2/build/sb_0-16513091-1442592600.21/mysql-5.6.27/storage/innobase/os/os0file.cc:5119
#3  0x0000000000a7732e in fil_aio_wait (segment=1) at /export/home/pb2/build/sb_0-16513091-1442592600.21/mysql-5.6.27/storage/innobase/fil/fil0fil.cc:5754
#4  0x00000000009dbbb8 in io_handler_thread (arg=<value optimized out>) at /export/home/pb2/build/sb_0-16513091-1442592600.21/mysql-5.6.27/storage/innobase/srv/srv0start.cc:489
#5  0x000000328fa079d1 in start_thread () from /lib64/libpthread.so.0
#6  0x000000328f6e8b6d in clone () from /lib64/libc.so.6

##
##  省略其他输出
##
```

> 使用 GDB 可以知道线程在做什么，但是无法和 MySQL5.7 版本中的 performance_schema.threads 表一样，可以把每个 线程ID（thread_os_id）和 线程NAME 对应起来。 MySQL5.6 中需要根据 GDB 的输出信息后，进行判断。

InnoDB（一）

1. InnoDB的历史

年份	事件	备注
1995	由Heikki Tuuri 创建Innobase Oy公司，并开发InnoDB存储引擎	Innobase开始做的是数据库，希望卖掉该公司
1996	MySQL 1.0 发布	
2000	MySQL3.23版本发布	
2001	InnoDB存储引擎集成到MySQL数据库	作为插件的方式集成
2006	Innobase被Oracle公司收购（InnoDB作为开源产品，性能和功能很强大）	InnoDB在被收购后的，MySQL中的InnoDB版本没有改变
2010	MySQL5.5版本InnoDB存储引擎称为默认存储引擎	MySQL被Sun收购，Oracle被Oracle收购，使得MySQL和InnoDB重新在一起配合开发
至今	其他存储引擎已经不再得到Oracle官方的后续开发	

2. InnoDB的特点

- Fully ACID（InnoDB默认的Repeat Read隔离级别就支持）
- Row-level Locking（支持行锁）
- Multi-version concurrency control（MVCC）（支持多版本并发控制）
- Foreign key support（支持外键）
- Automatic deadlock detection（死锁自动检测）
- High performance、High scalability、High availability（高性能，高扩展，高可用）

3. InnoDB存储引擎的文件

3.1 概述

InnoDB的文件主要分为两个部分，一个是表空间文件，一个是重做日志文件

- **表空间文件**
  - 独立表空间文件
  - 全局表空间文件
  - undo表空间文件（from MySQL5.6）
- **重做日志文件**
  - 物理逻辑日志
  - 没有Oracle的归档重做日志

3.2 InnoDB - 表空间

- **表空间的概念**
  - 表空间是一个 *逻辑存储* 的概念
  - 表空间可以由多个文件组成
  - 支持裸设备（可以直接使用 O_DIRECT 方式绕过缓存，直接写入磁盘）
- **表空间的分类**
  - **系统表空间** *（最早只有系统表空间）*
    - 存储元数据信息
    - 存储Change Buffer信息
    - 存储Undo信息
    - *甚至一开始 所有的表和索引 的信息都是存储在系统表空间中*
      - *随后InnoDB对其做了改进，可以使用独立的表空间*
  - **独立表空间**
    - `innodb-file-per-table=1` （开启支持每个表一个独立的表空间）
    - 每张用户表对应一个独立的 ibd文件
    - 分区表可以对应多个ibd文件
  - **Undo表空间**
    - MySQL5.6版本支持独立的Undo表空间
    - `innodb_undo_tablespaces`
  - **临时表空间**
    - MySQL5.7增加了临时表空间（ibtmp1）
    - `innodb_temp_data_file_path`

```
shell> ll ib*   # MySQL的datadir目录
-rw-r-----. 1 mysql mysql     22913 Dec 27 23:56 ib_buffer_pool
-rw-r-----. 1 mysql mysql  12582912 Jan  3 15:27 ibdata1         # 系统表空间，默认所有信息存在这里
-rw-r-----. 1 mysql mysql 134217728 Jan  3 15:27 ib_logfile0
-rw-r-----. 1 mysql mysql 134217728 Jan  3 15:27 ib_logfile1
-rw-r-----. 1 mysql mysql  12582912 Jan  3 15:27 ibtmp1          # 零时表空间
```

```
shell> cd burn_test # 在MySQL的datadir目录下，burn_test是自定义数据库，根据配置，默认为innodb的表
shell> ll test_1*
-rw-r-----. 1 mysql mysql  8554 Dec  3 20:14 test_1.frm #test_1表的表结构文件
                                               #mysqlfrm --diagnostic test_1.frm可查看表结构
-rw-r-----. 1 mysql mysql 49152 Dec  3 20:14 test_1.ibd #ibd就是test_1这张表对应的innodb文件
                                               #ibd中包含了索引和数据
```

> 同一个表空间（ibdata1） 存储和 独立表空间 存储就 性能 上而言没有区别；
>
> 当需要删除表（drop table）时， 独立的表空间 存储可以 直接删除文件 ，而 ibdata1 存储也只是把 该部分表空间标记为可用 ，所以从速度上看很难说哪个更快；但是 删除文件 后， ibdata1 占用的 空间不会释放 ；
>
> 分区表 会产生 独立 的 ibd文件 ；
>
> 独立的表空间 ，一个表对应一个 ibd文件 ，给人的感觉更加直观；
>
> **单个 ibd文件 直接拷贝到新的数据库中无法直接恢复：**
>
> 1. **原因一：** 元数据 **信息还是在 ibdata1 中**
> 2. **原因二：部分索引文件存在于 Change Buffer 中，目前还是存放于 ibdata1文件 中**

```
select * from information_schema.innodb_sys_tablespaces\G  -- 查看表空间的元数据信息
```

4. General表空间

> 官方文档

假如，新建一张表，并让该表的存储路径 不是默认的**/path/to/datadir/dbname** 。**而是** 指定存储的位置 **应该如何处理？**

- 方法一

```
shell> mkdir /GeneralTest
shell> chown mysql.mysql /GeneralTest


mysql> create table test_ger1 (a int) data directory='/GeneralTest';
Query OK, 0 rows affected (0.15 sec)


shell> cd /GeneralTest
shell> tree
.
└── burn_test                # dbname
    └── test_ger1.ibd        # 表空间文件

1 directory, 1 file

shell> ll  test_ger1*  # 在datadir 的 burn_test 目录下
-rw-r-----. 1 mysql mysql 8554 Jan  3 16:41 test_ger1.frm
-rw-r-----. 1 mysql mysql   36 Jan  3 16:41 test_ger1.isl  # 这是链接文件，链接到上面的ibd文件

shell> cat test_ger1.isl # 一个文本文件，内容就是idb文件的路径
/GeneralTest/burn_test/test_ger1.ibd
```

- 方法二

```
-- 使用`通用表空间`
-- 1: 创建一个通用表空间
mysql> create tablespace ger_space add datafile '/GeneralTest/ger_space.ibd' file_block_size=8192;
Query OK, 0 rows affected (0.07 sec)

-- datafile 指定存储路径后，在datadir下会产生一个isl文件，该文件的内容为General space的ibd文件的路径
-- 如果datafile不指定路径，则ibd文件默认存储在datadir目录下，且不需要isl文件了

mysql> create tablespace ger2_space add datafile 'ger2_space.ibd' file_block_size=8192;
Query OK, 0 rows affected (0.06 sec)


shell> ll ger*
-rw-r-----. 1 mysql mysql 32768 Jan  3 16:51 ger2_space.ibd  # 未指定路径，存放于datadir目录
-rw-r-----. 1 mysql mysql    26 Jan  3 16:50 ger_space.isl   # 指定了其他路径，存在isl链接文件
shell> cat ger_space.isl
/GeneralTest/ger_space.ibd   # ibd文件真实存在的路径


mysql> select * from information_schema.innodb_sys_tablespaces where name='ger_space'\G
*************************** 1. row ***************************
         SPACE: 96
          NAME: ger_space
          FLAG: 2304
   FILE_FORMAT: Any
    ROW_FORMAT: Any
     PAGE_SIZE: 8192      -- page_size是8k
 ZIP_PAGE_SIZE: 0
    SPACE_TYPE: General   -- General类型
 FS_BLOCK_SIZE: 0
     FILE_SIZE: 18446744073709551615
ALLOCATED_SIZE: 2
   COMPRESSION: None
1 row in set (0.00 sec)

-- 2: 创建表
mysql> create table test_ger2 (a int) tablespace=ger_space;
Query OK, 0 rows affected (0.11 sec)


shell> ll test_ger*       # 在datadir 的 burn_test 目录下
-rw-r-----. 1 mysql mysql 8554 Jan  3 16:41 test_ger1.frm
-rw-r-----. 1 mysql mysql   36 Jan  3 16:41 test_ger1.isl
-rw-r-----. 1 mysql mysql 8554 Jan  3 17:09 test_ger2.frm  # 仅有一个frm文件

shell> ll /GeneralTest/
total 52
drwxr-x---. 2 mysql mysql  4096 Jan  3 16:41 burn_test
-rw-r-----. 1 mysql mysql 49152 Jan  3 17:09 ger_space.ibd  # test_ger2的ibd文件其实存储在ger_space.ibd的通用表空间中


mysql> create table test_ger3 (a int) tablespace=ger_space;  -- test_ger3 也存放在ger_space.ibd中
Query OK, 0 rows affected (0.09 sec)
```

> **通过使用 General Space ，一个表空间可以对应多张表**
> 当对表进行 alter 等操作时，还是和原来一样，无需额外语法指定表空间位置
> *可以简单的理解为把多个表的ibd文件合并在一起了*

```
mysql> create tablespace ger3_space add datafile '/GeneralTest/ger3_space.ibd' file_block_size=4096;  -- 创建4K大小的General Space也是可以的
Query OK, 0 rows affected (0.06 sec)

-- 但是注意，如果设置了innodb_page_size,且大小不是file_block_size，那么在创建表的时候会报错
mysql> create table  test_ger (a int) tablespace=ger3_space;
ERROR 1478 (HY000): InnoDB: Tablespace `ger3_space` uses block size 4096 and cannot contain a table with physical page size 8192
-- 既然无法创建表，那应该在创建general space时就应该报错啊？
-- 后续涉及压缩表时可以使用
```

> **这里的 file_block_size 就是 page_size**

注意：需要考虑在使用General Space后，备份工具是否能够支持