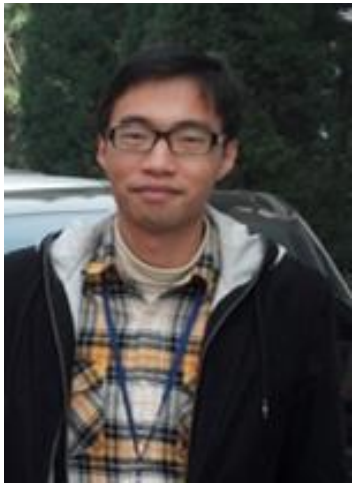

SH'OUG
SHANGHAI ORACLE USERS GROUP
上海ORACLE用户组

SH'OUG

MySQL故障诊断技巧 v0.1

by 汪伟华, Nov 2016

汪伟华



- 8年Oracle相关开发及数据库运维经验
  (Oracle DB, MySQL, Oracle Apps)
- MySQL OCP

ORACLE Certified Professional
MySQL 5.6 Database Administrator

SH'OUG
SHANGHAI ORACLE USERS GROUP
上海ORACLE用户组

Parnassus
诗檀

- E-mail：biot.wang@parnassusdata.com

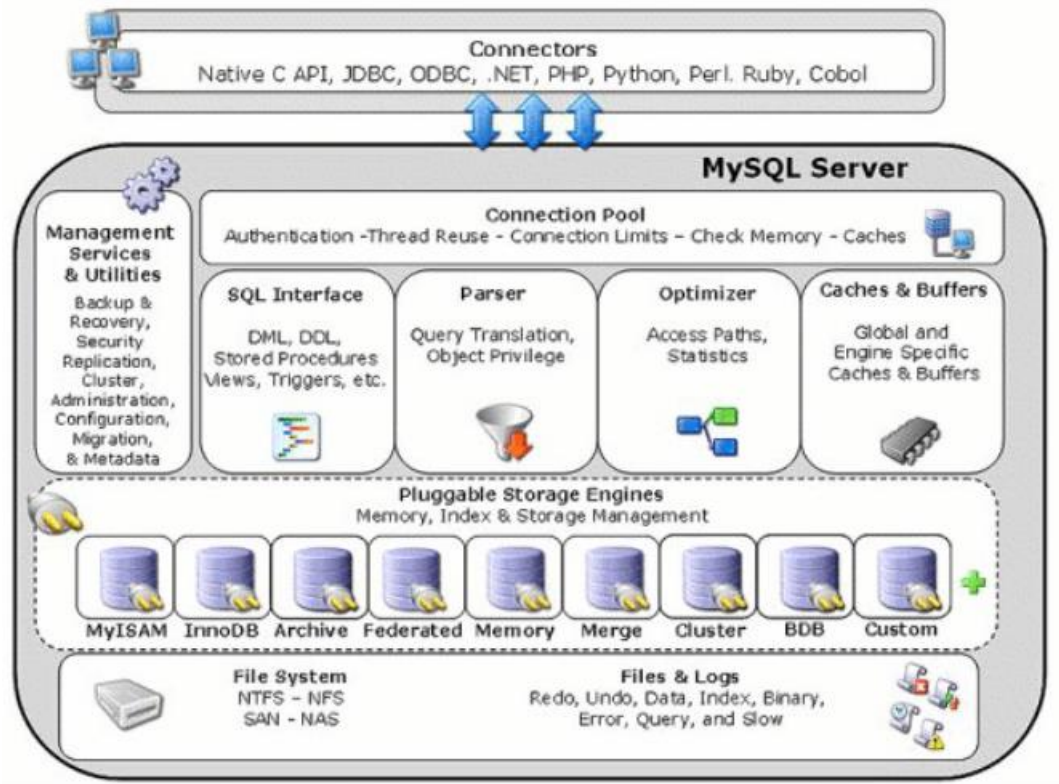
议程

- 基础回顾介绍
- 一般问题诊断
  - 一般单客户端问题
    - ✓ 配置项
    - ✓ 优化诊断
  - 并行问题
    - ✓ InnoDB监控器
    - ✓ P_S诊断
  - 高可用问题解决
- 工具
  - 对开发员
  - 对DBA

基础回顾介绍

总览

- 基础层
  - 安装文件层
  - 日志文件
- 连接器
  - 客户端
  - API等
- 优化器
- 缓存缓冲
- 存储引擎
- 管理



基础层

数据目录

- Schema
  - 表文件：*.frm，*ibd，*MYD，*MYI等
- 日志文件
- InnoDB共享表空间

日志文件

- 通用查询日志
- 慢查询日志
- 二进制日志（binary log）
- 中继日志（relay log）
- 报错日志
- InnoDB日志（以及其它存储引擎生成的日志）

连接器

客户端应用
- MySQL CLI 命令行接口
- MySQL Workbench
- MySQL Enterprise Monitor (MEM)
  - 管理员应用

APIs
- 对大多编程语言都已开发有相应接口程序
- C, C++, JDBC, PHP, Python, Net, ODBC等

插件（Plugin）

- 存储引擎
- 全文解析器（Full-text parser）
- 守护进程（Daemon）
- INFORMATION_SCHEMA
- 半同步复制（Semisynchronous Replication）
- 审计（Audit）
- 授权
- 密码验证

存储引擎

从**问题解决的角度**来看

- 自有数据及索引格式
- 自有锁模型
- 自有诊断
- 自有的日志文件
- CHECK TABLE

一般单客户端问题

MySQL访问权限系统

默认无角色和受限的用户限制设置
所有登陆权限相关记录存放在mysql schema下
从MySQL v5.5开始的可插拔授权

连接方式

- TCP/IP 密码登陆 / Socket (Unix) / Named pipe (Windows)

【Windows】
SELECT user, host FROM mysql.user;
SELECT USER(), CURRENT_USER();

多数情况会先选明确的host，然后才是 %

```
mysql> select user, host from mysql.user;
+-------+-----------+
| user  | host      |
+-------+-----------+
| root  | %         |
| root  | 127.0.0.1 |
| root  | ::1       |
| dbdao | localhost |
| root  | localhost |
+-------+-----------+
5 rows in set (0.06 sec)

mysql> select user(), current_user();
+----------------+----------------+
| user()         | current_user() |
+----------------+----------------+
| root@localhost | root@localhost |
+----------------+----------------+
1 row in set (0.03 sec)
```

MySQL访问权限系统 – 排序问题

```
mysql> select user, host from mysql.user
    ->  order by user desc, host desc;
+-------+-----------+
| user  | host      |
+-------+-----------+
| root  | localhost |
| root  | ::1       |
| root  | 127.0.0.1 |
| root  | %         |
| dbdao | localhost |
+-------+-----------+
5 rows in set (0.00 sec)
```

MySQL访问权限系统 – 访问出错与授权

SHOW GRANTS [FOR user@host]

授权表

- mysql.db
- mysql.tables_priv
- mysql.columns_priv
- mysql.procs_priv
- mysql.proxies_priv

```
mysql>
mysql> select user(), current_user();
+----------------+----------------+
| user()         | current_user() |
+----------------+----------------+
| root@localhost | root@localhost |
+----------------+----------------+
1 row in set (0.00 sec)

mysql> show grants;
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| Grants for root@localhost                                                                                                                     |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY PASSWORD '*DDB7BEFE01BCC8DC08
| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION
+-----------------------------------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)

mysql> show grants for dbdao@localhost;
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| Grants for dbdao@localhost                                                                                                                    |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'dbdao'@'localhost' IDENTIFIED BY PASSWORD '*DDB7BEFE01BCC8DC0
+-----------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

报错处理 - Errors vs warnings

报错 (Error) 和 告警 (warning)

```
mysql> select max (f1) from t1;
ERROR 1630 (42000): FUNCTION test.max does not exist. Check the
 'Function Name Parsing and Resolution' section in the Referenc
e Manual
mysql>
mysql> select * from t1 where "f1"=1;
Empty set, 1 warning (0.00 sec)

mysql> show warnings;
+---------+------+-----------------------------------------+
| Level   | Code | Message                                 |
+---------+------+-----------------------------------------+
| Warning | 1292 | Truncated incorrect DOUBLE value: 'f1'  |
+---------+------+-----------------------------------------+
1 row in set (0.00 sec)
```

应用端 (C API)的相应处理

报错信息

- mysql_errno
- mysql_error

告警信息

- mysql_info
- mysql_sqlstate
- mysql_warning_count

报错处理 – perror

```
root@dbdao-VirtualBox:~# perror 1630
MySQL error code 1630 (ER_FUNC_INEXISTENT_NAME_COLLISION): FUNCTION %s does not
exist. Check the 'Function Name Parsing and Resolution' section in the Reference Manual

root@dbdao-VirtualBox:~# perror 1292
MySQL error code 1292 (ER_TRUNCATED_WRONG_VALUE): Truncated incorrect %-.32s value: '%-
.128s‘

root@dbdao-VirtualBox:~# perror 2
OS error code   2:  No such file or directory

root@dbdao-VirtualBox:~# perror 150
MySQL error code 150: Foreign key constraint is incorrectly formed

root@dbdao-VirtualBox:~#
```

报错处理 – 存储例程

对于存储例程，使用 GET DIAGNOSTICS

GET DIAGNOSTICS rows = ROW_COUNT,
conditions = NUMBER;

GET DIAGNOSTICS CONDITION 1 code =
RETURNED_SQLSTATE,
msg = MESSAGE_TEXT;

http://dev.mysql.com/doc/refman/5.7/en/get-diagnostics.html

```
CREATE PROCEDURE do_insert(value INT)
BEGIN
  -- Declare variables to hold diagnostics area information
  DECLARE code CHAR(5) DEFAULT '00000';
  DECLARE msg TEXT;
  DECLARE rows INT;
  DECLARE result TEXT;
  -- Declare exception handler for failed insert
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
      GET DIAGNOSTICS CONDITION 1
        code = RETURNED_SQLSTATE, msg = MESSAGE_TEXT;
    END;

  -- Perform the insert
  INSERT INTO t1 (int_col) VALUES(value);
  …
END;
```

配置项

基于其全局或会话范围进行设置
- 可在配置文件中设置
- 可在命令行参数中设置
- SET [GLOBAL] var_name = NEW_VAL

仅内置两个级别的写限制
- SUPER权限可用于GLOBAL和部分有限的SESSION变量
- 非授权用户可以变更任意会话项设置！

会在以下情况下被分配
- 服务端启动时
- 用户连接建立时
- 对应某些操作

配置项 – 控制、查看

查看配置项状态

SHOW [GLOBAL] STATUS

GLOBAL
- 当服务端启动时

SESSION
- 当前会话操作
- 可以被重置
  - FLUSH STATUS

查看配置项全局状态

```
mysql> show global status like
'Handler_read_rnd_next';
+-----------------------------+-----------+
| Variable_name               | Value     |
+-----------------------------+-----------+
| Handler_read_rnd_next | 3821812 |
+-----------------------------+-----------+
1 row in set (0.00 sec)
```

查看配置项会话状态

```
mysql> show status like
'Handler_read_rnd_next';
+-----------------------------+---------+
| Variable_name               | Value |
+-----------------------------+---------+
| Handler_read_rnd_next | 8       |
+-----------------------------+---------+
1 row in set (0.00 sec)
```

配置项 – Troubleshooting 最佳实践

首先记录下当前的配置状态
- SHOW [GLOBAL] VARIABLES

动态修改配置项
- SET [GLOBAL] var_name=NEW_VAL

先在会话中进行测试

测试无误后，修改全局变量

没有问题后，修改配置文件对应配置

当你不知道哪个配置项出了问题时：

首先记录下当前的配置状态
- SHOW [GLOBAL] VARIABLES

启动mysqld时使用 --no-defaults 命令项
- 此命令项必须是在所有其它命令项之前，第一个

查看问题是否解决

一个接着一个修改变量进行尝试，直到发现问题配置所在

特定存储引擎

主要关心物理数据，所以数据信息问题都在此级别
- 讹误（Corruption）
- 索引统计信息

使用CHECK TABLE检查报错

MyISAM

数据存储在文件中
- *.frm – 表定义文件, 一般用于所有存储引擎
- *.MYD – 数据文件
- *.MYI – 索引文件

myisamchk --myisam_recovery_options在每次打开表时检查
- 第一次访问 – 表被打开
- 第二、第三、… 访问 – 使用已存在的描述符
- FLUSH TABLES – 关闭打开的描述符

InnoDB

事务存储引擎
物理层
- *.frm – 表定义文件
- 共享表空间
- *.ibd – 对独立表所存放的表空间
  - 可选设置项: --innodb_file_per_table
  - 推荐设置
- 重做日志文件

OPTIMIZE TABLE = ALTER + ANALYZE

对表自动启动检查

优化诊断

和Oracle相比, MySQL的EXPLAIN还不够强大

- 在 5.7.3 版本中有增强
- 在 MySQL Workbench 6.0实现了EXPLAIN的图形化
- MySQL EXPLAIN命令JSON输出新特性 [HOL9734, passed]

EXPLAIN EXTENDED

- 后跟SHOW WARNINGS信息

EXPLAIN PARTITIONS
EXPLAIN FORMAT=JSON

INFORMATION_SCHEMA.OPTIMIZER_TRACE
查看变量 'Handler_%' 状态

Oracle EXPLAIN

```
EXPLAIN PLAN SET statement_id = 'example_plan4'
FOR
  SELECT h.order_number, l.revenue_amount,
l.ordered_quantity
     FROM so_headers_all h, so_lines_all l
  WHERE h.customer_id = :b1
       AND h.date_ordered > SYSDATE30
       AND l.header_id = h.header_id ;

Plan
-----------------------------------------------------------------
SELECT STATEMENT
NESTED LOOPS
TABLE ACCESS BY INDEX ROWID SO_HEADERS_ALL
INDEX RANGE SCAN SO_HEADERS_N1
TABLE ACCESS BY INDEX ROWID SO_LINES_ALL
INDEX RANGE SCAN SO_LINES_N1
```

MySQL EXPLAIN

```
mysql> EXPLAIN SELECT user, host FROM user\G

*************************** 1. row ***************************
id: 1
select_type: SIMPLE
table: user
type: index
possible_keys: NULL
key: PRIMARY
key_len: 228
ref: NULL
rows: 4
Extra: Using index
1 row in set (0.00 sec)
```

EXPLAIN

```
mysql> explain select emp_no, salary from salaries order by (select max(salary) from salaries) \G
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: salaries
         type: ALL
possible_keys: NULL
          key: NULL
          ref: NULL
         rows: 2838525
     filtered: 100.00
        Extra: NULL
*************************** 2. row ***************************
           id: 2
  select_type: SUBQUERY
        table: salaries
         type: ALL
possible_keys: NULL
          key: NULL
          ref: NULL
         rows: 2838525
     filtered: 100.00
        Extra: NULL
2 rows in set, 1 warning (0.00 sec)
Note (Code 1003): /* select#1 */ select `employees`.`salaries`.`emp_no` AS `emp_no`,`employees`.`salaries`.`salary` AS `salary` from `employees`.`salaries` order by (/* select#2 */ select max(`employees`.`salaries`.`salary`) from `employees`.`salaries`)
```

EXPLAIN FORMAT=JSON

```
mysql> explain format=json select emp_no, salary from salaries order by (select max(salary) from salaries) \G
*************************** 1. row ***************************
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "573505.00"
    },
    "ordering_operation": {
      "using_filesort": false,
      "table": {
        "table_name": "salaries",
        <skipped>
      "optimized_away_subqueries": [
        {
          "dependent": false,
          "cacheable": true,
          "query_block": {
            "select_id": 2,
            "cost_info": {
              "query_cost": "573505.00"
            },
            "table": {
              "table_name": "salaries",
              "access_type": "ALL",
....
1 row in set, 1 warning (0.00 sec)
```

优化诊断 - Handler_% 状态变量

```
mysql> flush status;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SHOW STATUS LIKE 'Handler_%';
+----------------------------+-------+
| Variable_name              | Value |
+----------------------------+-------+
| Handler_commit             | 0     |
| Handler_delete             | 0     |
| Handler_discover           | 0     |
| Handler_prepare            | 0     |
| Handler_read_first         | 0     |
| Handler_read_key           | 0     |
| Handler_read_last          | 0     |
| Handler_read_next          | 0     |
| Handler_read_prev          | 0     |
| Handler_read_rnd           | 0     |
| Handler_read_rnd_next      | 0     |
| Handler_rollback           | 0     |
| Handler_savepoint          | 0     |
| Handler_savepoint_rollback | 0     |
| Handler_update             | 0     |
| Handler_write              | 0     |
+----------------------------+-------+
16 rows in set (0.00 sec)
```

```
mysql> select count(*) from employees
join titles using(emp_no) where
title='Senior Engineer';
+------------+
| count(*) |
+------------+
| 97750    |
+------------+
1 row in set (3.24 sec)
```

```
mysql> SHOW STATUS LIKE
'Handler_%';
+----------------------------+----------+
| Variable_name              | Value    |
+----------------------------+----------+
| Handler_commit             | 1        |
| Handler_delete             | 0        |
| Handler_discover           | 0        |
| Handler_prepare            | 0        |
| Handler_read_first         | 1        |
| Handler_read_key           | 300027   |
| Handler_read_last          | 0        |
| Handler_read_next          | 397774   |
| Handler_read_prev          | 0        |
| Handler_read_rnd           | 0        |
| Handler_read_rnd_next      | 0        |
...
```

优化诊断 - INFORMATION_SCHEMA.OPTIMIZER_TRACE

变量：

- optimizer_trace="enabled=on|off,one_line=off|on"
- optimizer_trace_features="greedy_search=on,range_optimizer=on,dynamic_range=on,repeated_subselect=on"
- optimizer_trace_limit=1
- optimizer_trace_max_mem_size=16384
- optimizer_trace_offset=1
- end_markers_in_json=0|1

启用trace(默认为关闭状态) ：

```
SET optimizer_trace="enabled=on";
SELECT <your query here>;
SELECT * FROM INFORMATION_SCHEMA.OPTIMIZER_TRACE;
```

更多查询...

完成追踪后，禁用它：

```
SET optimizer_trace="enabled=off";
```

http://dev.mysql.com/doc/internals/en/optimizer-tracing.html

优化诊断 - INFORMATION_SCHEMA.OPTIMIZER_TRACE

```
mysql> SELECT * FROM INFORMATION_SCHEMA.OPTIMIZER_TRACE\G
*************************** 1. row ***************************
QUERY: select emp_no, min(from_date) from titles
group by emp_no limit 10
TRACE: {
"steps": [
{
"join_preparation": {
"select#": 1,
"steps": [
{
"expanded_query": "/* select#1 */ select `titles`.`emp_no` AS
`emp_no`,min(`titles`.`from_date`) AS `min(from_date)` from `titles` group
by `titles`.`emp_no` limit 10"
```

优化诊断 - INFORMATION_SCHEMA.OPTIMIZER_TRACE

join_preparation
join_optimization
- table_dependencies
- rows_estimation
- considered_execution_plans
- attaching_conditions_to_tables
- clause_processing
- refine_plan
- reconsidering_access_paths_for_index_ordering

join_execution

并行问题

锁类型

MDL锁
表锁
行锁

读锁
- 块写

写锁
- 块读和写

事务和其相关的锁

服务端级别
- MDL锁

引擎级别
- 表锁
- 行锁

AUTOCOMMIT
- 支持

一般诊断方式

SHOW [FULL] PROCESSLIST
- 可列出当前所有运行连接的通用工具

SHOW ENGINE INNODB STATUS

INFORMATION SCHEMA
- PROCESSLIST
- InnoDB表

PERFORMANCE SCHEMA
- MDL锁

SHOW PROCESSLIST

```
mysql> show processlist;
+----+--------+-----------+---------+---------------+--------+---------------------------------+---------------------------------------+
| Id | User   | Host      | db      | Command       | Time   | State                           | Info                                  |
+----+--------+-----------+---------+---------------+--------+---------------------------------+---------------------------------------+
| 23 | biot   | localhost | test    | Query         | 3      | User sleep                      | select *, sleep(20) from tlog         |
| 24 | biot   | localhost | test    | Query         | 1      | Waiting for table level lock    | update tlog set ln=5 where ln=4       |
| 28 | root   | localhost | NULL    | Query         | 0      | NULL                            | show processlist                      |
+----+--------+-----------+---------+---------------+--------+---------------------------------+---------------------------------------+
3 rows in set (0.00 sec)
```

相应线程信息也可以在INFORMATION_SCHEMA.PROCESSLIST表中找到

但是，对于仍然没有具体的锁的信息

特定于InnoDB存储引擎的锁事件

事务和引擎级锁都是针对的引擎级别，存储引擎有其自己的诊断方式，当碰到问题时，你需要用到它们。

InnoDB监控器
- SHOW ENGINE INNODB STATUS
- 锁监控器

INFORMATION_SCHEMA表
- INNODB_TRX
- INNODB_LOCKS
- INNODB_LOCK_WAITS

InnoDB监控器 – SHOW ENGINE INNODB STATUS

```
------------------------
TRANSACTIONS
------------------------
Trx id counter 1B1C
Purge done for trx's n:o < 1B19 undo n:o < 0
History list length 189
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 0, not started
MySQL thread id 28, OS thread handle 0x7fa9301a0700, query
id 8653 localhost root
show engine innodb status
---TRANSACTION 1B1B, ACTIVE 6 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 376, 1 row lock(s)
MySQL thread id 31, OS thread handle 0x7fa9301e1700, query
id 8652 localhost biot Updating
update tbllog set `Employee ID`=2345 where `Log Number`=1
------ TRX HAS BEEN WAITING 6 SEC FOR THIS LOCK TO
BE GRANTED:
RECORD LOCKS space id 0 page no 81967 n bits 72 index
`PRIMARY` of table `test`.`tlog`
trx id 1B1B lock_mode X locks rec but not gap waiting
Record lock, heap no 6 PHYSICAL RECORD: n_fields 5;
compact format; info bits 0
0: len 4; hex 80000001; asc ;;
1: len 6; hex 000000001b19; asc ;;
2: len 7; hex 14000140340110; asc @4 ;;
3: len 4; hex 80003039; asc 09;;
4: SQL NULL;
---TRANSACTION 1B19, ACTIVE 787 sec
2 lock struct(s), heap size 376, 1 row lock(s), undo log
entries 1
MySQL thread id 32, OS thread handle 0x7fa930222700, query
id 8647 localhost biot
```

没看到针对事务持有锁的信息！

InnoDB监控器

MySQL 5.7.4之前

Pseudo-tables
将阶段性日志记录在报错日志中
- CREATE TABLE innodb_monitor(f1 int) ENGINE=INNODB;

锁监控
- 修改InnoDB监控器输出格式
- CREATE TABLE innodb_lock_monitor(f1 int) ENGINE=INNODB;

MySQL 5.7.4之后，CREATE TABLE建立方法被移除了

使用innodb_status_output和innodb_status_output_locks系统变量来启用 InnoDB Monitor和InnoDB Lock Monitor

http://dev.mysql.com/doc/refman/5.7/en/innodb-enabling-monitors.html

InnoDB锁监控

```
TRANSACTION 1B19, ACTIVE 1935 sec
2 lock struct(s), heap size 376, 1 row lock(s), undo log entries 1
MySQL thread id 32, OS thread handle 0x7fa930222700, query id 8647 localhost biot
TABLE LOCK table `test`.`tlog` trx id 1B19 lock mode IX
RECORD LOCKS space id 0 page no 81967 n bits 72 index `PRIMARY` of table
`test`.`tlog` trx id 1B19 lock_mode X locks rec but not gap
Record lock, heap no 6 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
0: len 4; hex 80000001; asc ;;
1: len 6; hex 000000001b19; asc ;;
2: len 7; hex 14000140340110; asc @4 ;;
3: len 4; hex 80003039; asc 09;;
4: SQL NULL;;
```

InnoDB Information Schema表

- INNODB_TRX
- INNODB_LOCKS
- INNODB_LOCK_WAITS

```
mysql> SELECT * FROM INNODB_LOCK_WAITS;
+-------------------+-------------------+-------------------+-------------------+
|requesting_trx_id|requested_lock_id|blocking_trx_id|blocking_lock_id|
+-------------------+-------------------+-------------------+-------------------+
| 1B1F              | 1B1F:0:81967:6    | 1B19              | 1B19:0:81967:6    |
+-------------------+-------------------+-------------------+-------------------+
1 row in set (0.00 sec)

mysql> select trx_mysql_thread_id, trx_id from innodb_trx;
+-----------------------+---------+
| trx_mysql_thread_id | trx_id |
+-----------------------+---------+
| 31                    | 1B1F   |
| 32                    | 1B19   |
+-----------------------+---------+
2 rows in set (0.00 sec)
```

通过P_S进行查看诊断

监控内部操作

- Events
- Waits
- Mutexes
- Statements
- Stages

类似于Oracle等待接口

Improving Performance with MySQL Performance Schema [HOL9733, Sunday, 1:00 PM]
Making the Performance Schema Easier to Use [CON5282, Sunday, 10:00 AM]
Performance Schema and ps_helper [CON4077, passed]

对于MDL锁的诊断查询

仅查看等待的线程
SHOW PROCESSLIST,
INFORMATION_SCHEMA.PROCESSLIST

- 对表元数据锁等待

Performance Schema

- MUTEX_INSTANCES
- EVENTS_WAITS_CURRENT
- THREADS

```
mysql> SELECT * FROM mutex_instances
           WHERE LOCKED_BY_THREAD_ID IS NOT NULL\G
*********************** 1. row ***********************
NAME:
wait/synch/mutex/sql/MDL_wait::LOCK_wait_status
OBJECT_INSTANCE_BEGIN: 37104744
LOCKED_BY_THREAD_ID: 18
1 row in set (0.01 sec)
```

```
mysql> SELECT THREAD_ID, EVENT_ID, EVENT_NAME, SOURCE, TIMER_START,
OBJECT_INSTANCE_BEGIN, OPERATION FROM events_waits_current WHERE
THREAD_ID IN(SELECT LOCKED_BY_THREAD_ID FROM mutex_instances WHERE
LOCKED_BY_THREAD_ID IS NOT NULL)\G
*************************** 1. row ***************************
THREAD_ID: 18
EVENT_ID: 461
EVENT_NAME: wait/synch/cond/sql/MDL_context::COND_wait_status
SOURCE: mdl.cc:1210
TIMER_START: 97623253523306
OBJECT_INSTANCE_BEGIN: 0
OPERATION: timed_wait
1 row in set (0.00 sec)
```

高可用问题解决

MySQL Cluster

特定的存储引擎: NDB
数据被存储在两个或多个物理机上
一般会有两个或更多拷贝

在故障分析诊断时
- 通常的troubleshooting技术都可用
- 特定NDB存储引擎的技术

MySQL Replication

易设置，总是可用。但如果要用异步（Asynchronous）master-slave，就必须在使用前进行设置启用

主库（Master）
- 将所有更新写入binary log

从库（Slave）
- IO线程会从主库读取更新并记录到relay log文件中
- SQL线程会将更新执行到备库上

Troubleshooting工具
- 从报错日志文件（Error log file）中分析
- Slave： SHOW SLAVE STATUS
- Master：SHOW MASTER STATUS
- mysqlbinlog

Replication IO thread: 通信问题

访问报错

- 查看从库报错日志
- SHOW SLAVE STATUS
- 尝试MySQL客户端通过从库登陆账号进行正常登陆连接
  - SHOW GRANTS
- 解决主库权限问题
- 重启从库

Replication SQL thread: 典型问题

简单的master-slave结构

- 主从数据不一致
  - 复制事件无法继续应用到从库
- 主从库出现不同报错
- 从库同步速度远跟不上主库

环形复制（Circular replication）或在从SQL thread有额外的写入

- 出现主从数据不一致

主备数据不一致

表除了SQL thread更新外是否还做了其它修改？

- 如何做的修改？
- 操作是否对表内容产生了负面影响?

主从库之间的表定义是否一致？

是否主库的操作事件并未以正确的顺序应用到从库？

- 可使用mysqlbinlog来查找引发问题的查询
- 检查主库应用来辅助发现问题

主库事件同步顺序出错

锁问题

- InnoDB

触发器

- SET GLOBAL slave_skip_counter
- 进行表同步!

使用不同配置项

- 使用主库配置项来启动从库, 从而检查问题

备库数据同步远落后于主库

线程

- 主库运行着多个更新线程
- 过去版本从库使用单个SQL thread，现在需要对slave_parallel_workers进行恰当设置

对从库性能进行调优

- Buffers
- 语句索引等

使用工具

对开发人员

MySQL Workbench
- SQL编辑器
- Database designer

MySQL命令行客户端
- SQL命令行接口
- **SELECT, INSERT, UPDATE, DELETE**, etc.

对DBA

MySQL Enterprise Monitor
- 实时MySQL性能和可用性监控
- 报错信息
- Advisor
- Forcast

MySQL Workbench
- Server administration
- Backups

MySQL命令行客户端
- SQL命令行接口
- **GRANT, CREATE, OPTIMIZE**, etc.

mysqladmin

Command line

可用于各种管理工作的工具

- MySQL Workbench Utilities
  http://dev.mysql.com/downloads/tools/utilities/
- Percona Toolkit
  http://www.percona.com/software/percona-toolkit

MySQL Sandbox

- 做测试用的Sandbox
  https://launchpad.net/mysql-sandbox

PS_HELPER视图

- www.markleith.co.uk/ps_helper/

更多参考

MySQL User Reference Manual
- http://dev.mysql.com/doc/refman/5.6/en/index.html

知识管理库

论坛
- http://forums.mysql.com

Bug追溯
- http://bugs.mysql.com
- Oracle Internal Bugs database

My Oracle Support
- https://support.oracle.com

MySQL Troubleshooting书籍
- http://shop.oreilly.com/product/0636920021964.do

Marc Alff's Performance Schema博客
- http://marcalff.blogspot.ru/

Planet MySQL
- http://planet.mysql.com/
- http://dev.mysql.com/support/blogs/
- https://blogs.oracle.com/mysqlinnodb/

Using, Learning & Sharing

SHOUG

Let's Leverage Oracle Together


SH'OUG
SHANGHAI ORACLE USERS GROUP
上海ORACLE用户组