由 OSC 引起的线上变更问题

written by tnt

问题描述

在日常发布任务时候，对一个表做 DDL 变更，由于有修改字段长度的操作，所以采用 pt-online-schema-change 变更，运行了大概十几秒之后，突然出现线上 active thread running 的报警，应用大量超时报警，然后 MHA 认为 DB 已经超时未响应，做了故障切换。

故障复原

故障现象如下图， 可以看见出现大量 replace into 堵塞语句

Cmd	ID	State	User	Host	DB	Time	Query
Query	131	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	132	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	133	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	134	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	135	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	136	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	137	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	143	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	145	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	146	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	147	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	169	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	172	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	177	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	221	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	248	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	274	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	300	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	329	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	437	update	autoddl	10.129.64.148	consulttimer	00:24	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	448	Sending data	autoddl	10.129.64.148	consulttimer	00:24	INSERT LOW_PRIORITY IGNORE INTO `consulttimer`.`_n
Query	645	update	autoddl	10.129.64.148	consulttimer	00:16	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	554	update	autoddl	10.129.64.148	consulttimer	00:14	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	596	update	autoddl	10.129.64.148	consulttimer	00:14	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	186	update	autoddl	10.129.64.148	consulttimer	00:12	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	388	update	autoddl	10.129.64.148	consulttimer	00:12	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	578	update	autoddl	10.129.64.148	consulttimer	00:12	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	614	update	autoddl	10.129.64.148	consulttimer	00:12	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	567	updating	consultt	10.129.165.130	consulttimer	00:11	update timeout_task set status = 2,owner_node = '
Query	161	update	autoddl	10.129.64.148	consulttimer	00:11	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	183	update	autoddl	10.129.64.148	consulttimer	00:11	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	568	update	autoddl	10.129.64.148	consulttimer	00:11	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	571	update	autoddl	10.129.64.148	consulttimer	00:11	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	576	update	autoddl	10.129.64.148	consulttimer	00:11	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	618	update	autoddl	10.129.64.148	consulttimer	00:11	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	623	update	autoddl	10.129.64.148	consulttimer	00:11	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	650	update	autoddl	10.129.64.148	consulttimer	00:11	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	139	update	autoddl	10.129.64.148	consulttimer	00:10	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	579	update	autoddl	10.129.64.148	consulttimer	00:10	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	619	update	autoddl	10.129.64.148	consulttimer	00:10	REPLACE INTO `consulttimer`.`_timeout_task_new` (
Query	635	update	autoddl	10.129.64.148	consulttimer	00:10	REPLACE INTO `consulttimer`.`_timeout_task_new`

故障研究

由于出现大量 replace into 堵塞，所以很可能是由于 trigger 导致的，为了简化验证环境，我们从原表，导出了 300 条记录到测试环境，并新建了表_timeout_task_new，同时再 timout_task 上建了 update，delete， insert 三个 trigger,结构如下

	源表	目标表
表结构	CREATE TABLE `timeout_task` (`id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT '自增主键', `out_biz_type` varchar(20) NOT NULL COMMENT '业务方', `out_biz_id` varchar(32) NOT NULL COMMENT '业务 id', `out_biz_key` varchar(32) NOT NULL COMMENT '业务 Key', `payload` varchar(2000) DEFAULT NULL COMMENT '负载参数', `action_time` datetime NOT NULL COMMENT '执行时间', `pause_time` datetime DEFAULT NULL COMMENT '暂停时间', `status` tinyint(4) NOT NULL DEFAULT '0' COMMENT '任务状态 代码取值: 0-未处理 1-已处理 2-取消 3-暂停', `owner_node` varchar(32) NOT NULL DEFAULT '' COMMENT '所属节点', `trigger_way` tinyint(1) NOT NULL DEFAULT '0' COMMENT '触发方式 代码取值: 0-MQ', `in_timer` tinyint(1) DEFAULT '0' COMMENT '是否在 timer 中 代码取值: 0-否 1-是', `gmt_created` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '记录创建时间', `gmt_modified` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '记录最后修改时间', `env` varchar(16) NOT NULL DEFAULT '' COMMENT '运行环境', PRIMARY KEY (`id`), UNIQUE KEY `uniq_biz_key` (`out_biz_id`,`out_biz_type`,`out_biz_key`), KEY `idx_action_time` (`action_time`), KEY `idx_gmt_created` (`gmt_created`), KEY `idx_owner_node_status_gmt_created`	CREATE TABLE `_timeout_task_new` (`id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT '自增主键', `out_biz_type` varchar(20) NOT NULL COMMENT '业务方', `out_biz_id` varchar(32) NOT NULL COMMENT '业务 id', `out_biz_key` varchar(32) NOT NULL COMMENT '业务 Key', `payload` varchar(2000) DEFAULT NULL COMMENT '负载参数', `action_time` datetime NOT NULL COMMENT '执行时间', `pause_time` datetime DEFAULT NULL COMMENT '暂停时间', `status` tinyint(4) NOT NULL DEFAULT '0' COMMENT '任务状态 代码取值: 0-未处理 1-已处理 2-取消 3-暂停', `owner_node` varchar(32) NOT NULL DEFAULT '' COMMENT '所属节点', `trigger_way` tinyint(1) NOT NULL DEFAULT '0' COMMENT '触发方式 代码取值: 0-MQ', `in_timer` tinyint(1) DEFAULT '0' COMMENT '是否在 timer 中 代码取值: 0-否 1-是', `gmt_created` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '记录创建时间', `gmt_modified` datetime NOT NULL DEFAULT

	(`owner_node`,`status`,`gmt_created`), KEY `idx_owner_node_status_action_time` (`owner_node`,`status`,`action_time`), KEY `idx_in_timer` (`in_timer`), KEY `idx_status_action_time` (`status`,`action_time`)) ENGINE=InnoDB AUTO_INCREMENT=15644286 DEFAULT CHARSET=utf8mb4 COMMENT='超时任务';	CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '记录最后修改时间', `env` varchar(16) NOT NULL DEFAULT '' COMMENT ' 运行环境', PRIMARY KEY (`id`), UNIQUE KEY `uniq_biz_key` (`out_biz_id`,`out_biz_type`,`out_biz_key`), KEY `idx_action_time` (`action_time`), KEY `idx_gmt_created` (`gmt_created`), KEY `idx_owner_node_status_gmt_created` (`owner_node`,`status`,`gmt_created`), KEY `idx_owner_node_status_action_time` (`owner_node`,`status`,`action_time`), KEY `idx_in_timer` (`in_timer`), KEY `idx_status_action_time` (`status`,`action_time`)) ENGINE=InnoDB AUTO_INCREMENT=10873307 DEFAULT CHARSET=utf8mb4 COMMENT='超时任务';
update	REPLACE INTO `test`.`_timeout_task_new` (`id`, `out_biz_type`, `out_biz_id`, `out_biz_key`, `payload`, `action_time`, `pause_time`, `status`, `owner_node`, `trigger_way`, `in_timer`, `gmt_created`, `gmt_modified`, `env`) VALUES (NEW.`id`, NEW.`out_biz_type`, NEW.`out_biz_id`, NEW.`out_biz_key`, NEW.`payload`, NEW.`action_time`, NEW.`pause_time`, NEW.`status`, NEW.`owner_node`, NEW.`trigger_way`, NEW.`in_timer`, NEW.`gmt_created`, NEW.`gmt_modified`, NEW.`env`)	
delete	DELETE IGNORE FROM `test`.`_timeout_task_new` WHERE `test`.`_timeout_task_new`.`id` <=> OLD.`id`	
insert	REPLACE INTO `test`.`_timeout_task_new` (`id`, `out_biz_type`, `out_biz_id`, `out_biz_key`, `payload`, `action_time`, `pause_time`, `status`, `owner_node`, `trigger_way`, `in_timer`, `gmt_created`, `gmt_modified`, `env`) VALUES (NEW.`id`,	

	NEW.`out_biz_type`, NEW.`out_biz_id`, NEW.`out_biz_key`, NEW.`payload`, NEW.`action_time`, NEW.`pause_time`, NEW.`status`, NEW.`owner_node`, NEW.`trigger_way`, NEW.`in_timer`, NEW.`gmt_created`, NEW.`gmt_modified`, NEW.`env`)	

场景 1：

Session 1	session 2

mysql> select count(*) from timeout_task;
+----------+
| count(*) |
+----------+
| 281 |
+----------+
1 row in set (0.00 sec)
mysql> select count(*) from _timeout_task_new;
+----------+
| count(*) |
+----------+
| 0 |
+----------+
1 row in set (0.00 sec)	

begin;	
mysql> update timeout_task set status = 0 where id = 10873306; Insert intention gap lock Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0	
	begin;

由于 session 1 和 session 2 不是插入的同一行记录，Insert intention lock 之间兼容，所以两者不会发生锁等待。	mysql> update timeout_task set status = 0 where id = 10873166; （没有锁等待） Insert intention gap lock Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0
commit;	rollback;

场景 2：

Session 1	Session 2
mysql> select id,out_biz_id from _timeout_task_new; +----------+------------+ | id | out_biz_id | +----------+------------+ | 10873306 | 171821886 | +----------+------------+ 1 row in set (0.00 sec) uniq_biz_key 的索引记录范围是 `test`.`_timeout_task_new` (-∞,171821886],(171821886,+∞)	mysql> select id,out_biz_id from timeout_task where id = 10873166; +----------+------------+ | id | out_biz_id | +----------+------------+ | 10873166 | 171820347 | +----------+------------+ 1 row in set (0.00 sec)
mysql> begin; Query OK, 0 rows affected (0.00 sec) mysql> update timeout_task set status = 0 where id = 10873306; Query OK, 0 rows affected (0.00 sec) Rows matched: 1 Changed: 0 Warnings: 0 unique key 记录存在，加 Share lock next key lock (-∞,171821886],(171821886,+∞) 成功，加 Insert intention lock (gap -∞ 到 +∞)	
场景一和二更新的唯一区别就是，场景一中`test`.`_timeout_task_new`表的 uniq_biz_key 的索引记录范围空，场景二中范围是(-∞,171821886],(171821886,+∞)，所以初步怀疑是有 uniqe key 的 gap lock	找一条 out_biz_id 小于 171821886 的记录

show engine innodb status 显示 session 2 在等待 _timeout_task_new 的 唯一索引 uniq_biz_key 的 gap lock 的插入意向锁,这是因为 session 1 已经获得 -∞ 到 +∞ Share next key lock, session 2 想要获得 insert intention lock, insert intention lock 与 next key lock 不兼容，所以 session 2 等待 RECORD LOCKS space id 99 page no 10 n bits 72 index `uniq_biz_key` of table `test`.`_timeout_task_new` trx id 36075550 lock_mode X locks gap before rec insert intention waiting 由于 171820347 < 171821886,所以是`test`.`_timeout_task_new`的索引 uniq_biz_key (-∞,171821886]的范围锁导致 session 2 blcok 住，那(171821886,+∞）是否也有 gap lock 呢？	mysql> begin; Query OK, 0 rows affected (0.00 sec) mysql> update timeout_task set status = 0 where id = 10873166; (锁等待) ^CCtrl-C -- sending "KILL QUERY 707046" to server ... Ctrl-C -- query aborted. ERROR 1317 (70100): Query execution was interrupted mysql> rollback; Query OK, 0 rows affected (0.00 sec)

show engine innodb status
显示 session 2 在等待 _timeout_task_new 的 唯一索引 uniq_biz_key 的 Next Key 插入意向锁，这是因为 session 1 已经获得 -∞ 到 +∞ Share next key lock, session 2 想要获得 insert intention lock, insert intention lock 与 next key lock 不兼容，所以 session 2 等待
RECORD LOCKS space id 99 page no 10 n bits 72 index `uniq_biz_key` of table `test`.`_timeout_task_new` trx id 36075554 lock_mode X insert intention waiting
结论：
唯一索引记录的前后范围都被锁住。
那么索引记录的前后范围的第一条索引记录是否也被锁住了？
	找一条 out_biz_id 大于 171821886 的记录
	mysql> select id,out_biz_id from timeout_task where id = 10873307;
	+----------+------------+
	| id | out_biz_id |
	+----------+------------+
	| 10873307 | 171822261 |
	+----------+------------+
	1 row in set (0.00 sec)
	begin;
	mysql> update timeout_task set status = 0 where id = 10873307; （锁等待）
	^CCtrl-C -- sending "KILL QUERY 707046" to server ...
	Ctrl-C -- query aborted.
	ERROR 1317 (70100): Query execution was interrupted

	mysql> rollback; Query OK, 0 rows affected (0.00 sec)

场景 3：

session 1	session 2
mysql> select id,out_biz_id from _timeout_task_new; +----------+------------+ | id | out_biz_id | +----------+------------+ | 10873306 | 171821886 | | 10873028 | 171822130 | +----------+------------+ 2 rows in set (0.00 sec) uniq_biz_key 的索引记录范围是 `test`.`_timeout_task_new` (- ∞,171821886],(171821886,171822130],(171822130,+∞)	mysql> select id,out_biz_id from timeout_task where id =10873027; +----------+------------+ | id | out_biz_id | +----------+------------+ | 10873027 | 171822124 | +----------+------------+ 1 row in set (0.00 sec)
mysql> begin; Query OK, 0 rows affected (0.00 sec) mysql> update timeout_task set status = 1 where id = 10873306; Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0 unique key 记录存在，加 (-∞,171821886],(171821886,171822130] Share next key lock 成功，加 Insert intention lock (gap -∞, 171822130)	
show engine innodb status 显示 session 2 在等待 _timeout_task_new 的 唯一索引 uniq_biz_key 的 gap lock 的插入意向锁 RECORD LOCKS space id 99 page no 10 n bits 72 index `uniq_biz_key` of table	mysql> begin; Query OK, 0 rows affected (0.00 sec) mysql> update timeout_task set status = 1 where id = 10873027;

`test`.`_timeout_task_new` trx id 36075570 lock_mode X locks gap before rec insert intention waiting	^CCtrl-C -- sending "KILL QUERY 707046" to server ... Ctrl-C -- query aborted. ERROR 1317 (70100): Query execution was interrupted mysql> rollback; Query OK, 0 rows affected (0.00 sec)
show engine innodb status 显示 session 2 在等待 _timeout_task_new 的 唯一索引 uniq_biz_key 的记录锁,No gap 锁，session 2 等待 next key lock 中的记录锁 RECORD LOCKS space id 99 page no 10 n bits 72 index `uniq_biz_key` of table `test`.`_timeout_task_new` trx id 36075571 lock_mode X locks rec but not gap waiting 得出结论 唯一索引 171821886 被锁住时候，(171821886,171822130]的 gap 和 record 都被锁住，下面看下索引记录的前面区间的第一条记录是否也会被锁住	mysql> begin; Query OK, 0 rows affected (0.01 sec) mysql> update timeout_task set status = 1 where id = 10873028; ^CCtrl-C -- sending "KILL QUERY 707046" to server ... Ctrl-C -- query aborted. ERROR 1317 (70100): Query execution was interrupted mysql> rollback; Query OK, 0 rows affected (0.00 sec)
mysql> rollback; Query OK, 0 rows affected (0.00 sec) mysql> select id,out_biz_id from _timeout_task_new; +----------+------------+ | id | out_biz_id | +----------+------------+ | 10873306 | 171821886 | | 10873028 | 171822130 | +----------+------------+ 2 rows in set (0.00 sec) mysql> begin; Query OK, 0 rows affected (0.00 sec) mysql> update timeout_task set status = 1 where id = 10873028; Query OK, 0 rows affected (0.00 sec) Rows matched: 1 Changed: 0 Warnings: 0	

unique key 记录存在，加(171821886,171822130],(171822130,+∞） Share next key lock 成功，加 Insert intention lock (gap 171821886 ，+∞)	
show engine innodb status 显示 session 2 在等待 _timeout_task_new 的 唯一索引 uniq_biz_key 的记录 Next key lock,因为 session 2 想要获得(-∞,171821886],(171821886,171822130]的 insert intention lock 但是和 session 2 的(171821886,171822130],(171822130,+∞） Share next key lock 冲突 RECORD LOCKS space id 99 page no 10 n bits 72 index `uniq_biz_key` of table `test`.`_timeout_task_new` trx id 36075580 lock_mode X waiting 这样得出结论当存在唯一索引记录(-∞,X],(X,Y],(Y,Z],(Z,+∞），当 Y 被锁住时候，**[X,Z]的前后闭区间，都会被锁住**	mysql> begin; Query OK, 0 rows affected (0.00 sec) mysql> update timeout_task set status = 11 where id = 10873306; ^CCtrl-C -- sending "KILL QUERY 707046" to server ... Ctrl-C -- query aborted. ERROR 1317 (70100): Query execution was interrupted mysql> rollback; Query OK, 0 rows affected (0.00 sec)

重现过程中还发现了在 read commited 隔离级别下，range in share mode 也会有 gap 锁存在

Session 1	Session 2
mysql> begin; Query OK, 0 rows affected (0.00 sec) mysql> update timeout_task set status = 11 where id = 10873306; Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0	
发下 session 2 在等待一个非 gap 的记录的共享锁 RECORD LOCKS space id 75 page no 13 n bits 152 index `PRIMARY` of table `test`.`timeout_task` trx id 36075599 lock mode S locks rec but not gap waiting	mysql> select * from timeout_task where id > 10873293 and id <=10873305 lock in share mode; ^CCtrl-C -- sending "KILL QUERY 733594" to server ... Ctrl-C -- query aborted. ERROR 1317 (70100): Query execution was interrupted

可以看出，再 lock in share mode 中，如果是 id 的 range 查询，其实也是需要拿到扫描范围的前后 gap 锁

```
mysql> select * from timeout_task where   id >
10873293 and id  <10873305  lock in share mode;
+----------+-------------+-----------+-------------
-------------+---------+--------------------+------
------+--------+---------------+------------+------
----+--------------------+--------------------+----
--+
| id           | out_biz_type | out_biz_id |
out_biz_key                        | payload |
action_time              | pause_time | status |
owner_node        | trigger_way | in_timer |
gmt_created              | gmt_modified            |
env   |
+----------+-------------+-----------+-------------
-------------+---------+--------------------+------
------+--------+---------------+------------+------
----+--------------------+--------------------+----
--+
| 10873294 | CONSULT         | 171822257   |
10_2TYn3I5YBfabTQccV1LadD | NULL      | 2018-08-14
08:10:06 | NULL           |         0 |
10.129.165.140 |                 0 |           0 |
2018-08-14 07:55:05 | 2018-08-22 19:09:13 | PROD |
```

这样可知，当时线上变更的问题，原因是：随着 osc 变更的时间增加，_timeout_task_new 表中的记录越来越多，唯一索引的区间段增加，由于线上有显示提交事务的 update 语句，这样就有可能造成场景二和场景三的锁等待问题，而一旦线上出现锁等待语句，由于 insert trigger 产生的插入_timeout_task_new 语句的自增字段的 mutex 全部锁升级为表级别的 auto_inc table lock,这样造成 osc 的批量 copy 语句 insert into _timeout_task_new xxx select * from timeout_task where id > xx and id <xx in share mode 无法获得 _timeout_task_new 的 auto_inc table lock，而这个时候线上业务依然在继续更新，后续所有更新产生的 trigger 语句全都被 hang 住，最终影响了业务表 timeout_task。

参考：https://dev.mysql.com/doc/refman/5.6/en/innodb-auto-increment-handling.html

```
LATEST DETECTED DEADLOCK
------------------------
2018-08-21 14:11:55 7f0962e39700TOO DEEP OR LONG SEARCH IN THE LOCK TABLE WAITS-FOR GRAPH, WE WILL ROLL BACK FOLLOWING
TRANSACTION
*** TRANSACTION:
TRANSACTION 145332107, ACTIVE 2 sec setting auto-inc lock
mysql tables in use 2, locked 2
4 lock struct(s), heap size 1184, 1 row lock(s), undo log entries 2
MySQL thread id 600, OS thread handle 0x7f0962e39700, query id 142812 10.129.165.130 consulttimer update
REPLACE INTO `consulttimer`.`_timeout_task_new` (`id`, `out_biz_type`, `out_biz_id`, `out_biz_key`, `payload`,
`action_time`, `pause_time`, `status`, `owner_node`, `trigger_way`, `in_timer`, `gmt_created`, `gmt_modified`, `env`)
VALUES (NEW.`id`, NEW.`out_biz_type`, NEW.`out_biz_id`, NEW.`out_biz_key`, NEW.`payload`, NEW.`action_time`,
NEW.`pause_time`, NEW.`status`, NEW.`owner_node`, NEW.`trigger_way`, NEW.`in_timer`, NEW.`gmt_created`, NEW.`gmt_modified`,
NEW.`env`)
*** WAITING FOR THIS LOCK TO BE GRANTED:
TABLE LOCK table `consulttimer`.`_timeout_task_new` trx id 145332107 lock mode AUTO-INC waiting
*** WE ROLL BACK TRANSACTION (2)
------------
```