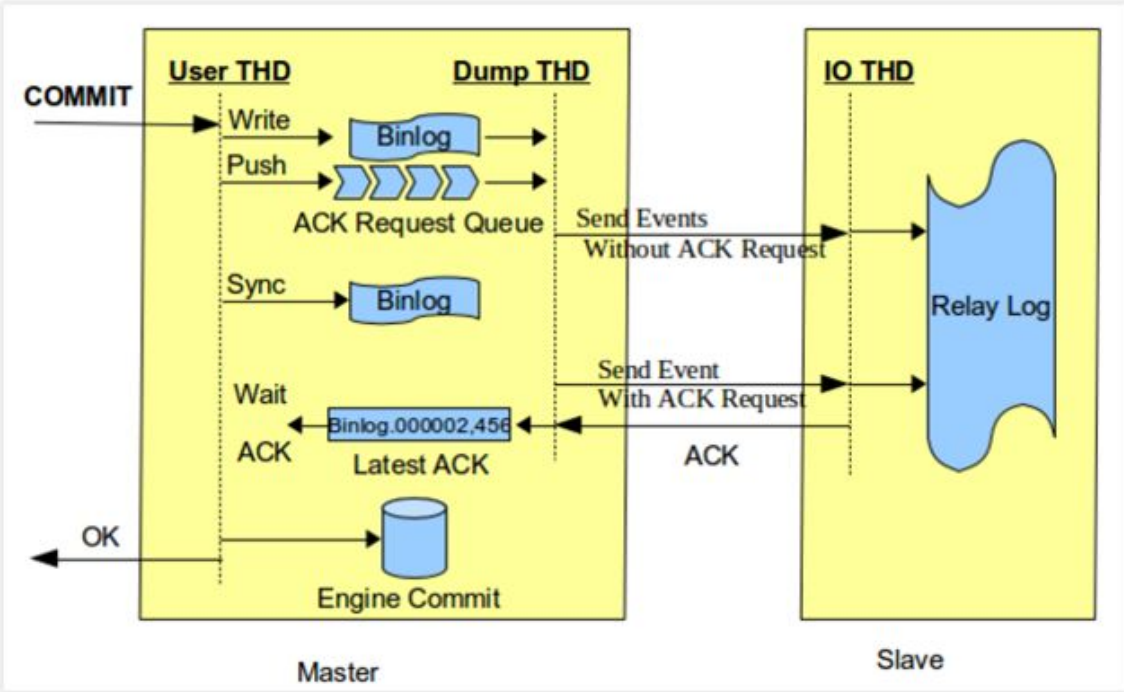
Phantom-read problem in semi-sync replication

1. Bug description

Following is a loss-less semi-sync principle picture from

http://my-replication-life.blogspot.co.uk/2013/09/loss-less-semi-synchronous-replication.html



The bug can be recured in a cluster with rpl_semi_sync_master_wait_slave_count > 0 if master restart before "Engine Commit" phase. During the restart, the last binlog events will be committed without slave's ack. A phantom-read problem can be recurrenced now:

Any clients will read this events successfully on master. But these clients will not read them on slave if master crash down in a very small period.(slave is switched into master manually or automatically).

This problem can also be found on the above article:

To make the crashed master server before MySQL 5.7.2 to work again, users need to:

1. Manually truncate the binlog events which are not replicated.
2. Manually rollback the transactions which are committed by not replicated.

Since this feature guarantees all committed transactions are replicated already, so 2nd step is not needed any more.

2. Bug recurrence

1. Install semi-sync on both master and slave.

Master:

```
| INNODB_SYS_DATAFILES          | ACTIVE | INFORMATION SCHEMA | NULL                | GPL     |
| INNODB_CHANGED_PAGES          | ACTIVE | INFORMATION SCHEMA | NULL                | GPL     |
| partition                     | ACTIVE | STORAGE ENGINE     | NULL                | GPL     |
| rpl_semi_sync_master          | ACTIVE | REPLICATION        | semisync_master.so  | GPL     |
```

Slave:

```
| INNODB_CHANGED_PAGES          | ACTIVE | INFORMATION SCHEMA | NULL              | GPL     |
| partition                     | ACTIVE | STORAGE ENGINE     | NULL              | GPL     |
| rpl_semi_sync_slave           | ACTIVE | REPLICATION        | semisync_slave.so | GPL     |
+-------------------------------+--------+--------------------+-------------------+---------+
```

2. Execute ‘Insert into bug_table values(1);’

Master status:

```
mysql> create table bug_table ( c1 int );
Query OK, 0 rows affected (0.04 sec)

mysql> insert into bug_table values(1);
Query OK, 1 row affected (0.00 sec)

mysql> select * from bug_table;
+------+
| c1   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
```

Slave status:

```
Database changed
mysql> select * from bug_table;
+------+
| c1   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
```

3. client connects to master and sends ‘Insert into bug_table values(2);’

(we made master restarted while executing this SQL(after binlog is written)).

```
mysql> Insert into bug_table values(2);
ERROR 2013 (HY000): Lost connection to MySQL server during query
mysql>
```

4.client reconnect to master and execute ‘select * from bug_table;’

```
Database changed
mysql> select * from bug_table;
+------+
| c1   |
+------+
|    1 |
|    2 |
+------+
2 rows in set (0.00 sec)
```

Here we can see value 2 was been inserted in Storage Engine.

5.kill mysql again and makes it always dead.

```
ERROR 2003 (HY000): Can’t connect to MySQL server on ’10.121.105.161’ (111)
```

4. Now, client have to read from slave, execute ‘select * from bug_table’ on slave.

```
mysql> select * from bug_table;
+------+
| c1   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
```

The result shows client meets a phantom-read problem.

3. Solution suggestion

Offer a new hook point to solve this problem.

A)A hook point after binlog opened.

Semi-sync could wait for the slave ack while recovery.

B)A hook point before binlog initilization.

Rollback the binlog event if necessary.

4. PhxSQL as a sample.

We implemented solution B in PhxSQL cluster

https://github.com/tencent-wechat/phxsql/tree/master/phx_percona/percona/sql

```
class Binlog_storage_delegate
  :public Delegate {
public:

  Binlog_storage_delegate()
  : Delegate(
#ifdef HAVE_PSI_INTERFACE
             key_rwlock_Binlog_storage_delegate_lock
#endif
             )
  {}

  typedef Binlog_storage_observer Observer;
  int after_flush(THD *thd,
                  const char *dir_path,
                  const char *prev_log_file,
                  my_off_t prev_log_pos,
                  const char *log_file,
                  my_off_t log_pos);

  int before_binlog_init(THD *thd, const char * server_uuid,
                         PSI_file_key * key_file_binlog_index,
                         const char * log_bin_index );
};
```

We do hope this solution be accepted officially.