Advanced MySQL replication techniques

Giuseppe Maxia
About me

http://datacharmer.org
Agenda

• Replication basics
• Circular replication
• Automatic failover
• Failover with circular replication
Replication goals

• Data redundancy
• Load balancing
• Backup points
• Base for failover
Replication basics

- One single point of data insertion and changes (MASTER)
- More points of data read (SLAVES)
- Different IDs for each server
- Binary log
- Replication slave user
- Replication threads (IO and SQL)
Replication basics

- Replication basics
- Binary log
- Relay log
- Read (I/O)
- Run (SQL)

Diagram showing the replication process involving master, slave 2, and slave 3.
Circular Replication

- More points of data insertion and changes (MASTER)
- Each slave is also a master
- Insertion conflicts!
Circular Replication

Diagram showing two nodes, node A and node D, with IP addresses 172.16.1.10 and 172.16.1.20, respectively. The diagram indicates data replication between these nodes, with 'water' and 'air' labels on each node.
CREATE TABLE x (
    id int(11) NOT NULL AUTO_INCREMENT,
    c char(10) DEFAULT NULL,
    PRIMARY KEY (id)
) ENGINE=MyISAM
Circular Replication

# CONFLICT
#(assume broken connection between nodes)

[node A] insert into x (c) values ('aaa'), ('bbb'), ('ccc');

[node B] insert into x (c) values ('xxx'), ('yyy'), ('zzz');
Circular Replication

# CONFLICT
#(when connection is resumed)

Error 'Duplicate entry 'l' for key 'PRIMARY'' on query. Default database: 'test'. Query: 'insert into x (c) values ('aaa')'
### Circular Replication

**[node A]**
```
select *
from x;
```
```
+-------+
<table>
<thead>
<tr>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
+-------+
```

**[node B]**
```
select *
from x;
```
```
+-------+
<table>
<thead>
<tr>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
+-------+
```

*Table: x*

<table>
<thead>
<tr>
<th>id</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>aaa</td>
</tr>
<tr>
<td>2</td>
<td>bbb</td>
</tr>
<tr>
<td>3</td>
<td>ccc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>xxx</td>
</tr>
<tr>
<td>2</td>
<td>yyy</td>
</tr>
<tr>
<td>3</td>
<td>zzz</td>
</tr>
</tbody>
</table>

**[node A]**
```
select *
from x;
```
```
+-------+
<table>
<thead>
<tr>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
+-------+
```

**[node B]**
```
select *
from x;
```
```
+-------+
<table>
<thead>
<tr>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
+-------+
```

*Table: x*

<table>
<thead>
<tr>
<th>id</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>xxx</td>
</tr>
<tr>
<td>2</td>
<td>yyy</td>
</tr>
<tr>
<td>3</td>
<td>zzz</td>
</tr>
</tbody>
</table>
Circular Replication
solving auto-increment conflicts
Circular Replication

# SOLVING AUTO-INCREMENT CONFLICTS
# in both nodes

STOP SLAVE;
RESET MASTER;
RESET SLAVE;
TRUNCATE x;
Circular Replication

# SOLVING AUTO-INCREMENT CONFLICTS

[NodeA]
set auto_increment_increment = 10;
set auto_increment_offset = 1;

[NodeB]
set auto_increment_increment = 10;
set auto_increment_offset = 2;
Circular Replication

# SOLVING AUTO-INCREMENT CONFLICTS
# (Slaves still stopped)

[node A] insert into x (c) values ('aaa'), ('bbb'), ('ccc');

[node B] insert into x (c) values ('xxx'), ('yyy'), ('zzz');
**Circular Replication**

[node A] select * from x;

<table>
<thead>
<tr>
<th>id</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>aaa</td>
</tr>
<tr>
<td>11</td>
<td>bbb</td>
</tr>
<tr>
<td>21</td>
<td>ccc</td>
</tr>
</tbody>
</table>

[node B] select * from x;

<table>
<thead>
<tr>
<th>id</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>xxx</td>
</tr>
<tr>
<td>12</td>
<td>yyy</td>
</tr>
<tr>
<td>22</td>
<td>zzz</td>
</tr>
</tbody>
</table>
Circular Replication

# SOLVING AUTO-INCREMETN CONFLICTS
# (resume replication)

[node A] SLAVE START;

[node B] SLAVE START;
Circular Replication

# SOLVING AUTO-INCREMENT CONFLICTS
# (resume replication)

[node A] SHOW SLAVE STATUS \G

[node B] SHOW SLAVE STATUS \G

... Slave_IO_Running: Yes
Slave_SQL_Running: Yes

...
Failover basics

• Master replacement
• inform each slave about the new master
Failover scenario (2a)
Failover scenario (2b)

# in the master

CREATE TABLE who (server_id int)
ENGINE=MyIsam
Failover scenario (2c)

# in each slave

CREATE TABLE master_who (  
  server_id int  
) ENGINE=Federated  
CONNECTION='mysql://user:pwd@172.16.1.10:3306/replica/who';
Failover scenario (2d)

# in each slave (requires MySQL 5.1)
create event check_master_conn
  on schedule every 30 second
  enable
  do call check_master();
Failover scenario (2e)

create procedure check_master()
begin
  declare master_dead boolean default false;
  declare curx cursor for
    select server_id
    from replica.master_who;
  declare continue handler
    for SQLSTATE 'HY000'
    set master_dead = true;
...

Failover scenario (2f)

procedure check_master()
...

open curx;
# a failure to open the cursor
# occurs here
# setting the master_dead variable
# to true
Failover scenario (2g)

```
procedure check_master()
...
if (master_dead) then
    stop slave;
    change master to
        master_host='172.16.1.40',
        master_log_file='binlog_name',
        master_log_pos=0;
    start slave;
    alter event check_master_conn
disable;
end if;
```
Failover scenario (3)
Failover scenario (4)

CHANGE MASTER TO
master_host='172.16.1.40'
Circular Failover (1)

Replication flow
Circular Failover (2)
Circular Failover (3)
Try it!

- The Replication playground
  http://sourceforge.net/projects/my-repl-play
Failover example (1)

# replication playground

$ ./start_all.sh

$ ./check_slaves.sh

  Slave_IO_Running: Yes
  Slave_SQL_Running: Yes
  Slave_IO_Running: Yes
  Slave_SQL_Running: Yes
  Slave_IO_Running: Yes
  Slave_SQL_Running: Yes
  Slave_IO_Running: Yes
  Slave_SQL_Running: Yes
  Slave_IO_Running: Yes
  Slave_SQL_Running: Yes
Failover example (2)

# replication playground

$ ./use.sh my.nodeC.cnf replica
nodeC> show slave status \G
...

    Master_Host: 127.0.0.1
    Master_User: nodeCuser
    Master_Port: 10011

...
# replication playground

```sql
nodeC> select * from check_master_log;
```

<table>
<thead>
<tr>
<th>ts</th>
<th>master_status</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006-10-29 10:39:26</td>
<td>master OK</td>
</tr>
<tr>
<td>2006-10-29 10:39:56</td>
<td>master OK</td>
</tr>
</tbody>
</table>
Failover example (4)

# replication playground

$ ./stop_all.sh A
nodeA shutdown
Failover example (5)

# replication playground
	nodeC> select * from check_master_log;

+---------------------------------------------------------------+---------------+
| ts                       | master_status            |
+---------------------------------------------------------------+---------------+
| 2006-10-29 10:39:26     | master OK              |
| 2006-10-29 10:39:56     | master OK              |
| 2006-10-29 10:40:26     | master is dead         |
+---------------------------------------------------------------+
Failover example (6)

# replication playground

$ ./use.sh my.nodeC.cnf replica
nodeC> show slave status \G
...

    Master_Host: 127.0.0.1
    Master_User: nodeCuser
    Master_Port: 10021

...
THANKS
Any questions?
http://datacharmer.org