Advance Replication Monitoring

Gerardo “Gerry” Narvaja
@seattlegaucho
Agenda

- Short Introduction
  - Make sure we all speak the same language
- Scenarios
  - What can go wrong and why it may be OK
- What To Look For / At
  - What the variables mean
  - Some pretty pictures
- Conclusion
Introduction

- What happens in the master …

- … in the slave it becomes …

- Replication is **single-threaded**
  - IO Thread + **SQL Thread**
  - No contention in the slave, it should run faster
Most Basic Monitoring

- **SHOW SLAVE STATUS**
  - IO Thread
    - Usually flags communication issues
  - SQL Thread
    - Usually flags data related issues

- Application code
  - Maatkit: mk-heartbeat
    - Simple monitoring can be implemented at the shell
  - Implement your own heartbeat table
    - Can be used to measure quality of data on the slaves

- If you don't have this basic monitoring in place, is like taking backups and not testing restores.
SHOW SLAVE STATUS\G

Slave_IO_State: Waiting for master to send event
Master_Host: 10.55.197.108
Master_User: repl
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: mysql-bin.000447
Read_Master_Log_Pos: 673847271
Relay_Log_File: relay-bin.005771
Relay_Log_Pos: 673847416
Relay_Master_Log_File: mysql-bin.000447
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table: mysql.user,mysql.columns_priv,mysql.tables_priv,mysql.db,mysql.procs_priv,mysql.host
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 0
Last_Error: 0
Skip_Counter: 0
Exec_Master_Log_Pos: 673847271
Relay_Log_Space: 673847506
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Errno: 0
Last_IO_Error: 0
Last_SQL_Errno: 0
Last_SQL_Error: 0

IO thread health status
SQL thread health status
General health status
- What happens when storing BLOBs and loading them in batches

- SBC is based on the timestamp for the transaction
  - You can get crazy values based on the actual traffic
  - Is this a bad situation?
  - How do `master_log_file` and `read_master_log_pos` look like?
Not provided directly
- On the master: `SHOW MASTER STATUS, SHOW BINARY LOGS`

```
show master status; show binary logs;
```

```
+------------------+-----------+--------------+------------------+
| File             | Position  | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+-----------+--------------+------------------+
| mysql-bin.009734 | 153545495 |              |                  |
+------------------+-----------+--------------+------------------+
```

```
<table>
<thead>
<tr>
<th>Log_name</th>
<th>File_size</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql-bin.009730</td>
<td>1073764076</td>
</tr>
<tr>
<td>mysql-bin.009731</td>
<td>1073772807</td>
</tr>
<tr>
<td>mysql-bin.009732</td>
<td>1073761932</td>
</tr>
<tr>
<td>mysql-bin.009733</td>
<td>1073756776</td>
</tr>
<tr>
<td>mysql-bin.009734</td>
<td>153545495</td>
</tr>
</tbody>
</table>
```

- On the slave: `SHOW SLAVE STATUS`

Challenges
- Not easy way to get information from the master, but only need past files info
- Master position is a moving target
- ROW vs STATEMENT vs MIXED replication
- Example: Data purges → DELETE ... FROM table WHERE ...
Replication Capacity Index

- Based on **Estimating Replication Capacity** blog by Percona
  - Estimate the capacity of the slave to keep up with the master load
- Some bash scripts and real data
  - `#!/bin/bash`
    
    ```bash
    # Test RCI (Replication Capacity Index)
    echo "$(date +%Y%m%d-%H%M%S) - Starting test"
    mysql -e "stop slave"
    sleep 600
    mysql -e "start slave"
    ```
  - `while true; do`
    ```bash
    echo $(date +%Y%m%d-%H%M%S) - `mysql -e "show slave status\G" | grep -i seconds` >> test.log
    sleep 10
    done
    ```
- 20100729-205134 - Seconds_Behind_Master: 0
  20100729-205140 - Starting test --> Initial timestamp
  20100729-205144 - Seconds_Behind_Master: NULL

... 20100729-210134 - Seconds_Behind_Master: NULL
  20100729-210144 - Seconds_Behind_Master: 161
  20100729-210154 - Seconds_Behind_Master: 0 --> Last timestamp

<table>
<thead>
<tr>
<th></th>
<th>Pause</th>
<th>Start TS</th>
<th>1st TS</th>
<th>SBM</th>
<th>2nd TS</th>
<th>Diff 1</th>
<th>Diff 2</th>
<th>RCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>044</td>
<td>00:10:00</td>
<td>20:51:40</td>
<td>21:01:44</td>
<td>161</td>
<td>21:01:54</td>
<td>00:10:04</td>
<td>00:10:14</td>
<td>43.9</td>
</tr>
<tr>
<td>045</td>
<td>00:10:00</td>
<td>17:32:13</td>
<td>17:42:17</td>
<td>320</td>
<td>17:42:27</td>
<td>00:10:04</td>
<td>00:10:14</td>
<td>43.9</td>
</tr>
<tr>
<td>005</td>
<td>00:10:00</td>
<td>15:37:12</td>
<td>15:47:21</td>
<td>441</td>
<td>15:47:41</td>
<td>00:10:09</td>
<td>00:10:29</td>
<td>21.7</td>
</tr>
<tr>
<td>001</td>
<td>00:10:00</td>
<td>18:54:28</td>
<td>19:04:33</td>
<td>520</td>
<td>19:04:53</td>
<td>00:10:05</td>
<td>00:10:25</td>
<td>25.0</td>
</tr>
<tr>
<td>002</td>
<td>00:10:00</td>
<td>18:02:32</td>
<td>18:12:39</td>
<td>389</td>
<td>18:12:49</td>
<td>00:10:07</td>
<td>00:10:17</td>
<td>36.3</td>
</tr>
</tbody>
</table>
- Revisiting the replication delay chart
  - Lt: Time while replication falls behind
  - Rt: Time it takes for replication to catch up
  - RCI = Rt/Lt
Replication Heartbeat

- Using Maatkit's mk-heartbeat
  - Run on the active master with -update option
  - Run on the slaves with -monitor or --check option
  - Output similar to Linux' `uptime`

```bash
mk-heartbeat --monitor --host localhost --database maatkit
18s [ 2.85s, 0.57s, 0.10s ]
19s [ 3.17s, 0.63s, 0.21s ]
20s [ 3.50s, 0.78s, 0.23s ]
18s [ 3.80s, 0.76s, 0.25s ]
16s [ 4.07s, 0.81s, 0.27s ]
```

- Issues
  - Highly sensitive to clocks in the master and slave(s) being in sync
  - It has to run on the active master in master-to-master setups
  - Better than seconds behind master
How To Monitor?

- There is no silver bullet
  - Avoid noise alerts
- Know your monitoring system
  - Tools: OpenNMS (SNMP), MONyog, MySQL Enterprise, home grown
  - Don't rely on just one
- Alarms
  - Thresholds and hysteresis
  - Number of incidents until it alarms
  - Sampling intervals
- Know your load
  - Low / High traffic? Bursts?
  - Small / big transactions? Concurrency?
- Replication type
  - Row / Statement / Mixed
Thank you very much