PALOMINODB

Proven Database Excellence

Keeping a MySQL Slave in Sync Using mk-table-checksum and mk-table-sync

Presented by:
Sheeri K. Cabral
Database Operations Manager
www.palominodb.com

Why is a slave out of sync?

- Writes to slave
- Starting slave at wrong binlog position
- Statement-based replication
- Non-deterministic writes

Why is a slave out of sync?

- Slave restarts and in-memory tables
- Dependencies on non-replicated data
- Non-unique server_id
- Hard to pinpoint:
  - Network errors
  - MySQL bugs

Sync'ing the Slave

- Find out-of-sync tables
- Use mk-table-checksum on master

```
perl mk-table-checksum localhost -u user --ask-pass \
--replicate maatkit.checksum
```

- You can also specify chunks

mk-table-checksum

- First run, use --create-replicate-table

```
perl mk-table-checksum  localhost -u user --ask-pass \
--replicate maatkit.checksum --create-replicate-table
```

- Uses numeric/date/time index
  - Or locks the whole table (older versions, prior to Aug 2010 version 6839)

Chunking and Modulo

- Chunks use numeric indexes
- Modulo and offset

```
perl mk-table-checksum localhost -u user -ask-pass \
--replicate maatkit.checksum \
--chunk-size 1000 --modulo 21 --offset 7
```

- Chunks may not be exact
- No warning if no numeric index!

Chunking and Filtering

- --since and --since-column

```
perl mk-table-checksum localhost -u user -ask-pass \
--replicate maatkit.checksum --tables db.foo,db.bar \
--chunk-size 1000 --since "NOW()-INTERVAL 1 HOUR" \
--since-column modified_date
```

- --where

```
perl mk-table-checksum localhost -u user -ask-pass \
--replicate maatkit.checksum --tables db.foo,db.bar \
--chunk-size 1000
--where "modified_date>NOW()-INTERVAL 1 HOUR"
```


Find Out-of-Sync Data

- Query the slave:

```
SELECT db, tbl, boundaries, master_cnt, this_cnt, ts
FROM maatkit.checksum
WHERE this_crc!=master_crc AND db!='maatkit';
```

- boundaries = WHERE clause

Find Out-of-Sync Data

```
mysql> SELECT * FROM maatkit.checksum

WHERE this_crc!=master_crc\G

*************************** 1. row ***************************
          db: production
         tbl: foo
       chunk: 1176
  boundaries: `id` >= 11841145 AND `id` < 11851214
    this_crc: 9139bf59
    this_cnt: 295
  master_crc: f91afa73
  master_cnt: 245
          ts: 2010-10-08 16:23:40
```

Continual Replication Sync Checking

- Dynamic offset

```
perl mk-table-checksum localhost -u user -ppassword \
--replicate maatkit.checksum --create-replicate-table \
--modulo 21 --chunk-size 1000
--offset 'modulo_offset FROM maatkit.checksum_modulo'
```

- Run from cron
  - Look on the slave(s) regularly for differences
  - Can make a monitoring script

Now What?

- Why is the data out of sync?
  - Find differences
  - Use mysqldump on master and slave

    --skip-extended-insert
  - --where “id >= 11841145 AND id < 11851214”
  - Use diff to find differences

Now What?

- Fix systemic issues
  - --sysdate-is-now
  - Rogue scripts
  - Deterministic info
  - Use row-based/mixed binlogging if needed

Fix the data

- By hand if it is easy
  - e.g. update tbl set timefield=timefield + interval 3 hour where 1=1;
- Use a backup if it is too difficult
  - e.g. lots of foreign key dependencies

Fix the data with mk-table-sync

- Can run on slave or master
  - --verbose
  - --print
  - --replicate

```
perl /usr/bin/mk-table-sync --print --verbose \
--replicate heartbeat.checksum \
h=master_host,u=user,p=pass > runme.sql
```

Fix the data with mk-table-sync

- How I do it:

```
perl /usr/bin/mk-table-sync --print --execute --verbose \
--replicate heartbeat.checksum \
h=master_host,u=user,p=pass > thisWasDone.sql
```

Ways to use mk-table-sync

- --replicate without --sync-to-master
  - Master is DSN, find slaves and sync them
- --replicate with --sync-to-master
  - Run on slave, will connect to master
- --sync-to-master with one DSN
  - Run on slave, will connect to master
- More than one DSN
  - First DSN is master, rest are slaves

Questions? Comments?