MySQL Replikation
Neue Features in 5.5 und 5.6

DOAG SIG-MySQL 2013, München

Oli Sennhauser

Senior MySQL Consultant, FromDual GmbH

oli.sennhauser@fromdual.com

Über FromDual GmbH

- FromDual bietet neutral und unabhängig:
  - Beratung für MySQL und Galera
  - Support für MySQL und Galera
  - Remote-DBA Dienstleistungen
  - MySQL Schulungen
- Partner der Open Database Alliance (ODBA.org)
- Oracle Silver Partner (OPN)

www.fromdual.com

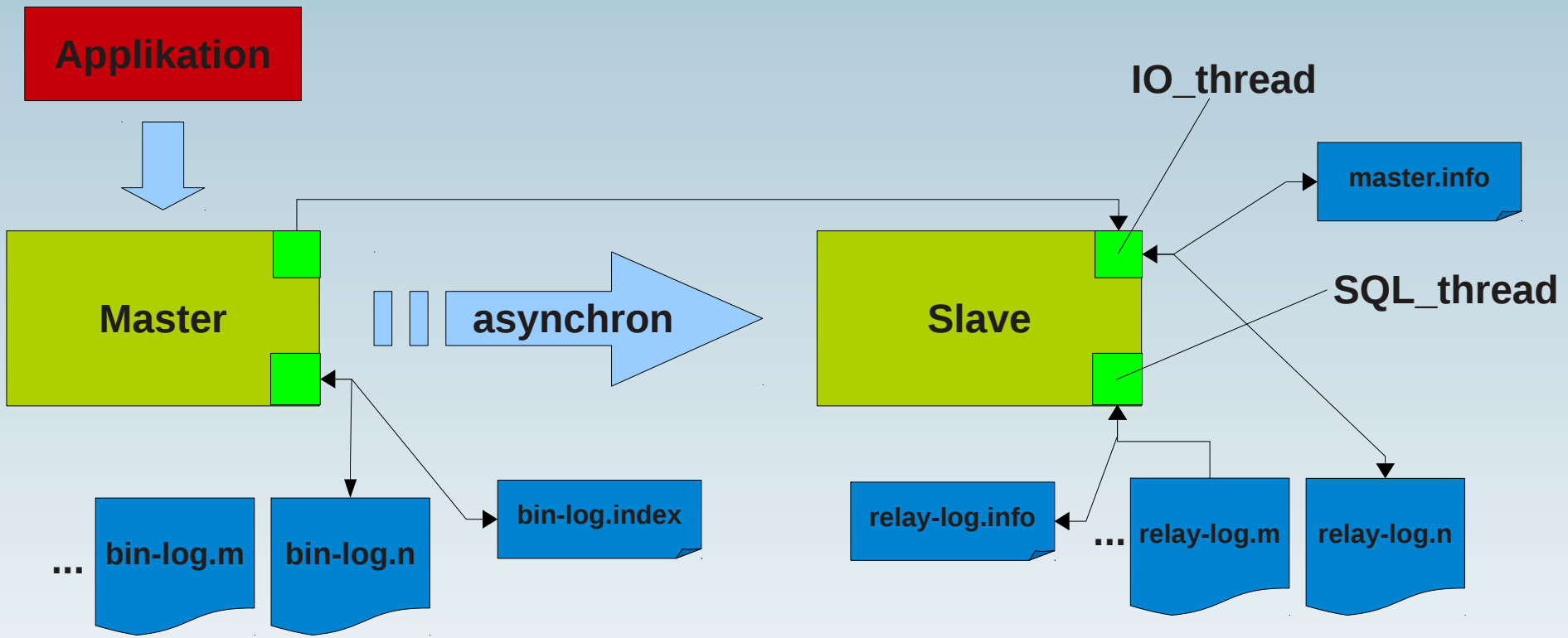
Inhalt

MySQL Replikation 5.5 & 5.6

- MySQL Replikation
- Binlog Format
- Row Image Control (RBR)
- Semi-synchrone Replikation
- Robustheit der MySQL Replikation
- Transaktionsbasierte Replikation
- Multi-threaded Slaves
- Weitere neue Features

MySQL Replikation

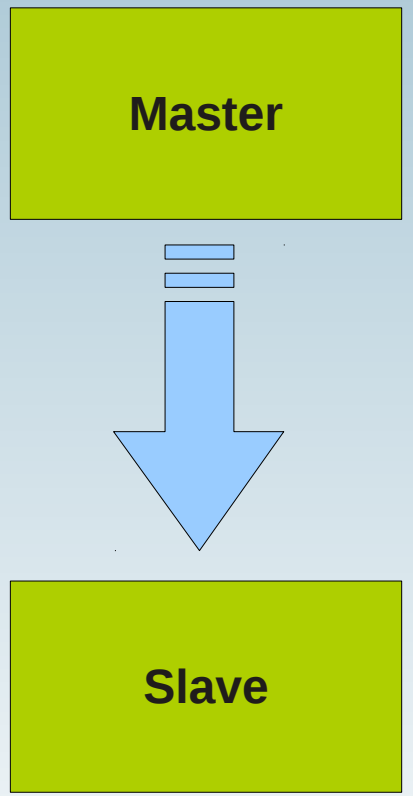
- **Wie funktioniert die MySQL Replikation?**



Binary Log Format

- MySQL <= 5.0 Statement Basiert (SBR)

```
# at 448
SET INSERT_ID=88925;
use `test`;
SET TIMESTAMP=1361802099;
INSERT INTO test
VALUES (NULL, 'Row inserted', NULL);
```



- MySQL >= 5.1 Row Basiert (RBR)

```
# at 245
BINLOG 'EnMrURMaiwAANQAAAPUAAAAAAE8AAAAAA
AEABHRlc3QABHRlc3QABAMPEQMDwAAACgGKfFI=
EnMrUR4aiwAAPQAAADIBAAAAAE8AAAAAAEAA
gAE//BcWwEADFJvdyBpbnNlcnRlZFErcxJ7AAAA
18yUQQ==';
```

Problem bei SBR

- Inkonsistenzen zwischen Master und Slave:
  - Nicht-deterministische Abfragen:

```
UPDATE emp SET sal = sal + 10 LIMIT 3;
```

- Was wird gemacht?
  - Auf Master?
  - Auf Slave?

→ Row Basierte Replikation (RBR)

Problem bei RBR

- `UPDATE email`
  `SET last_read = CURRENT_TIMESTAMP()`
  `WHERE id = 42;`
- Was passiert mit Mail Body oder Anhängen?
  - Bei SBR?
  - Bei RBR?

→ Row Image Control

Row Image Control

- Nur bei RBR
- Verringert:
  - Plattenplatz
  - Netzwerk Traffic
  - Speicher

```
binlog_row_image = {full | minimal | noblob}
```

binlog_row_image

- binlog_format = 'statement'

```
# at 207
UPDATE email SET ts = CURRENT_TIMESTAMP() WHERE id = 3
# at 343 (-207 = 136)
```

- binlog_row_image = 'full'

```
# at 454
BINLOG 'Hhgr ... k00=';
# at 640 (-454 = 186)
```

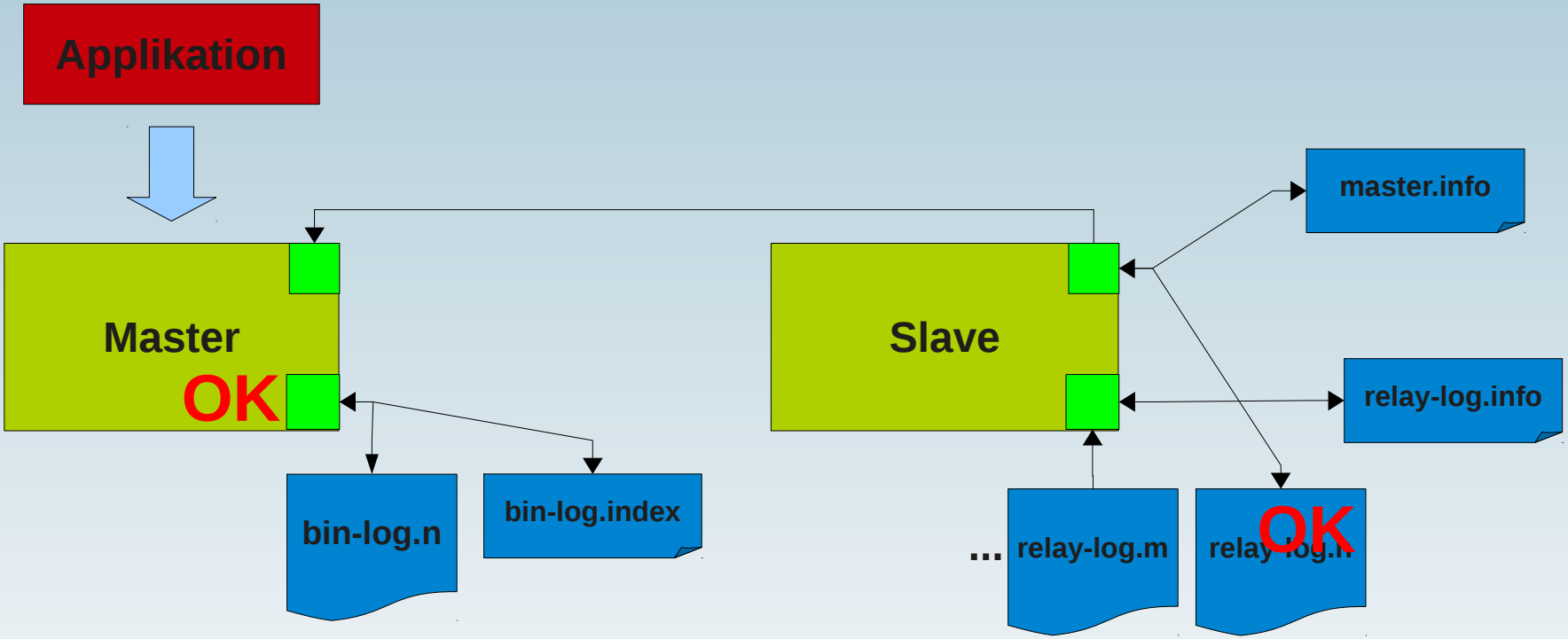
- binlog_row_image = 'minimal'

```
# at 751
BINLOG 'Jngr ... SA==';
# at 849 (-751 = 98)
```

- binlog_row_image = 'noblob'

Semi-synchrone Replikation

- **Wie funktioniert die semi-synchrone Replikation?**



Semi-synchrone Replikation (5.5)

- Default asynchrone Replikation (<= 5.1)
  - Master wartet NICHT auf Slave!
  - Bei Crash: Trx ist nicht zwingend auf Slave
- Neu (5.5) optional semi-synchrone Replikation
  - Plug-in (muss auf Master UND Slave aktiv sein!)
  - Master wartet auf Slave bis Timeout!
    - Nach Timeout (default 10 s) → Fallback auf asynchron
  - Bis Slave in Relay Log (sync) geschrieben hat
  - Bessere Datenintegrität (Master + mind. 1 Slave)
  - Schlechtere Performance (Commit + NW Roundtrip + Commit)
  - Master Commit, dann Crash, möglich dass Trx Slave nicht erreicht hat!

Semi-synchrone Replikation

- Plug-ins aktivieren:

```
INSTALL PLUGIN rpl_semi_sync_master SONAME 'semisync_master.so';
INSTALL PLUGIN rpl_semi_sync_slave SONAME 'semisync_slave.so';
```

- Prüfen ob erfolgreich:

```
SHOW PLUGINS;
+----------------------+--------+--------------------+-------------------+
| Name                 | Status | Type               | Library           |
+----------------------+--------+--------------------+-------------------+
| rpl_semi_sync_master | ACTIVE | REPLICATION        | semisync_master.so |
| rpl_semi_sync_slave  | ACTIVE | REPLICATION        | semisync_slave.so  |
+----------------------+--------+--------------------+-------------------+
```

- Semi-synchrone Replikation einschalten:

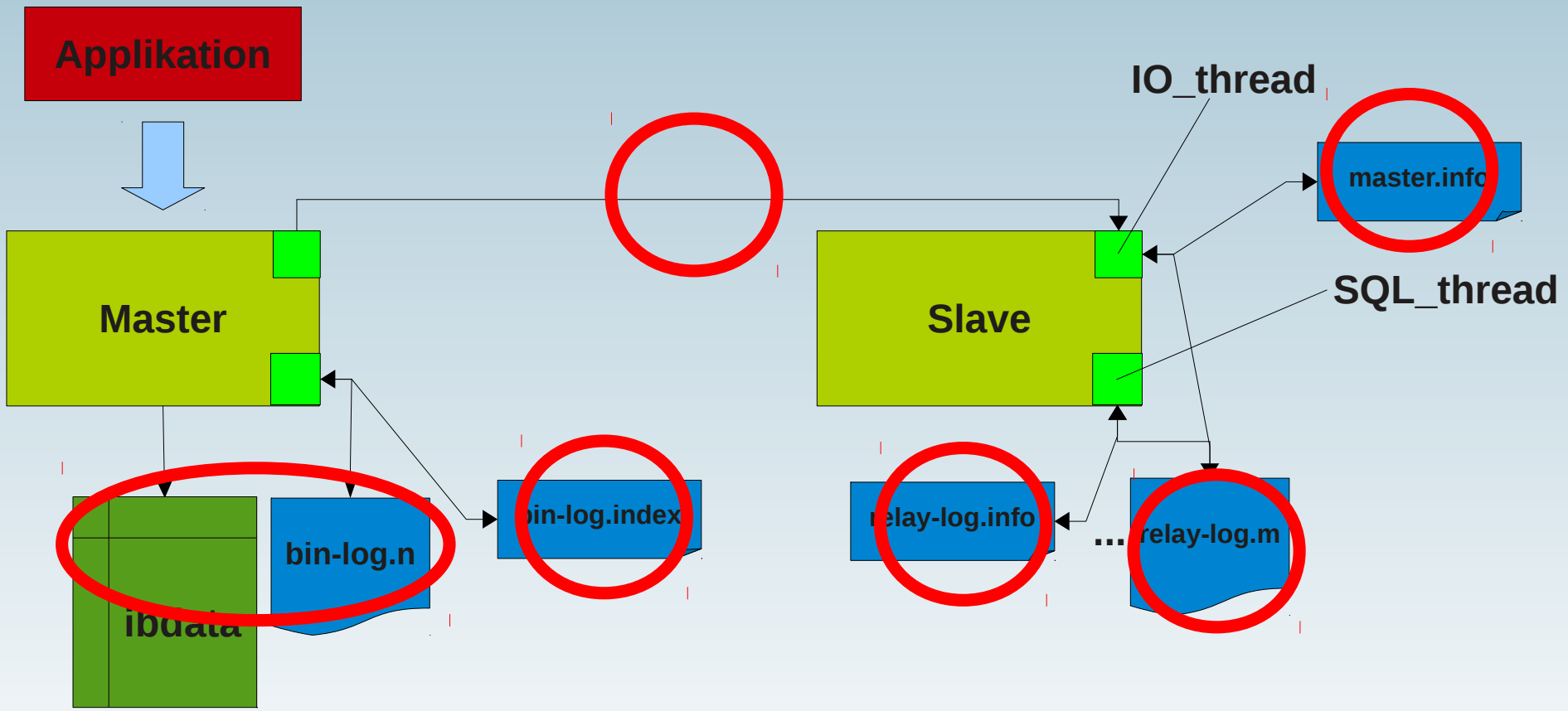
```
SET GLOBAL rpl_semi_sync_master_enabled = 1;
SET GLOBAL rpl_semi_sync_slave_enabled  = 1;
```

- Slave (neu) starten:

```
STOP SLAVE IO_THREAD; START SLAVE IO_THREAD;
```

Robustheit

- Was kann alles kaputt gehen?



Transaction Based Replikation

- Globally Unique Server ID
- Binary Log Group Commit (Performance)
- Vollständige Trx werden geschrieben/gelesen
- Replikations-Checksummen
- Globale Transaktion ID's (GTID)
- Replikations-Informationen in Tabellen
- Crash-safe Slaves

Globally Unique Server ID

- Wir automatisch erstellt:

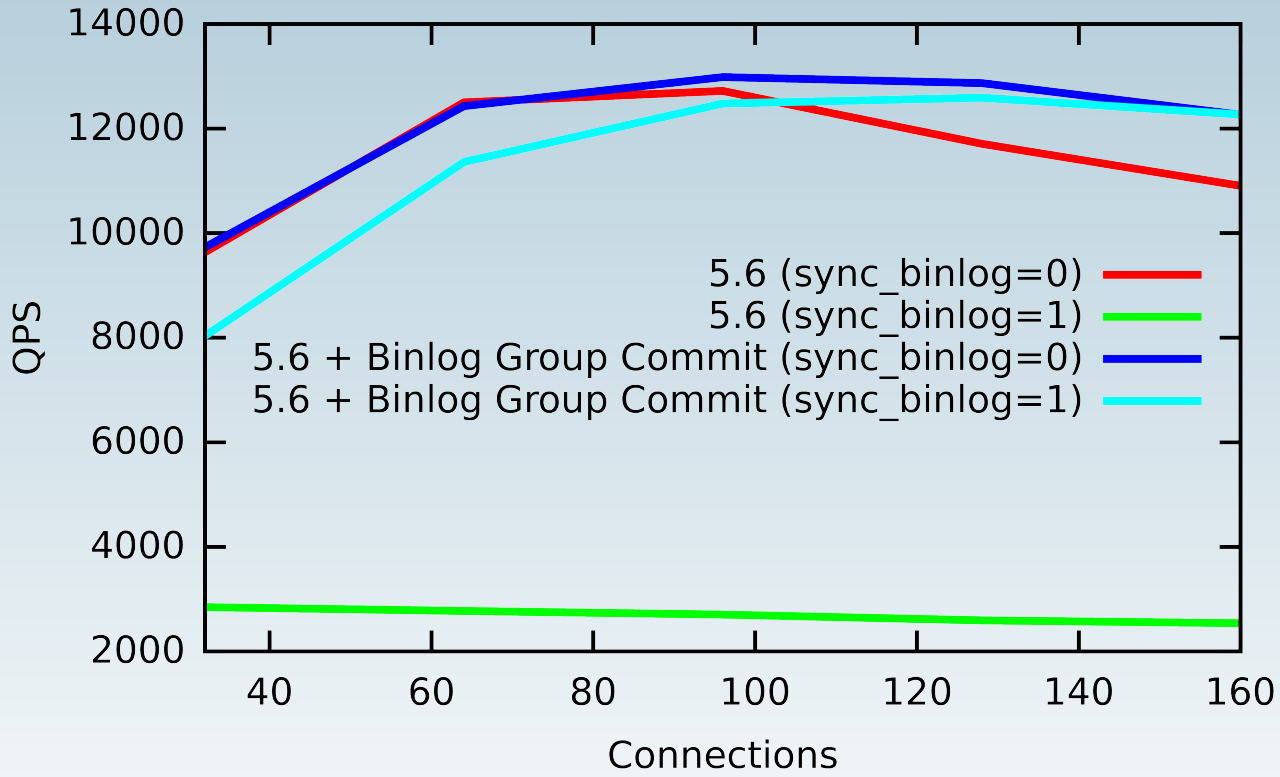
```
cat $datadir/auto.cnf
[auto]
server-uuid=63a3a056-f72f-11e1-840c-bcaec586ca65
```

```
SHOW GLOBAL VARIABLES LIKE 'server_uuid';
+---------------+--------------------------------------+
| Variable_name | Value                                |
+---------------+--------------------------------------+
| server_uuid   | 63a3a056-f72f-11e1-840c-bcaec586ca65 |
+---------------+--------------------------------------+
```

- Master und Slave sind eindeutig identifizierbar.

Binary Log Group Commit

- `sync_binlog = 1`



Crash-safe Binary Logs

- Nur vollständige Events/Trx werden geschrieben/gelesen
- Event-Länge + CRC32 Checksumme
  - Default: Länge + Event

```
SHOW GLOBAL VARIABLES LIKE '%checksum';

+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| binlog_checksum          | CRC32 |
| master_verify_checksum   | OFF   |
| slave_sql_verify_checksum | ON    |
+--------------------------+-------+
```

Global Transaction Identifiers

- Seit 5.6 GTID's:

  GTID = server_uuid:transaction_id

```
# at 448
# 130226  8:43:56 server id 35610  end_log_pos 496
# CRC32 0x966ee04b       GTID [commit=yes]
SET @@SESSION.GTID_NEXT='0e680a6a-e2b1-11e1-be68-bcaec586ca65:8';
```

- Auf Master UND Slave:

```
log_bin                  = binary-log
binlog_format            = ROW?
gtid_mode                = ON
log_slave_updates        = 1
enforce_gtid_consistency = 1
```

Slave mit GTID

- Konsistenten Snaphshot einspielen

```
CHANGE MASTER TO
  MASTER_HOST            = '192.168.1.42'
, MASTER_PORT            = 3306
, MASTER_USER            = 'replication'
, MASTER_PASSWORD        = 'secret'
, MASTER_AUTO_POSITION = 1;
```

```
START SLAVE;
```

- Unter „auto“ verstehe ich was anderes...

Crash-safe Slave

- Replikationsinformation in Tabelle:

  `master.info` → `slave_master_info`

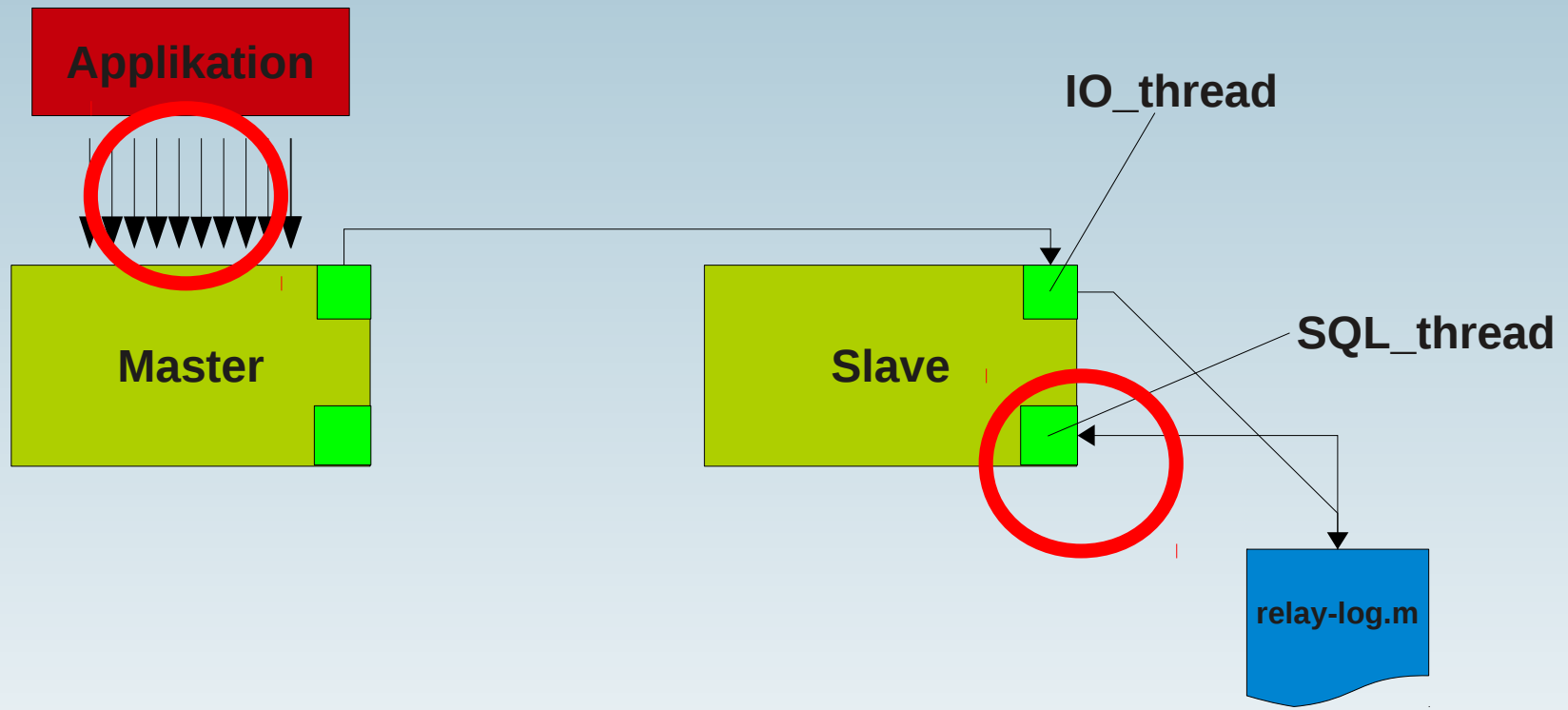
  `relay-log.info` → `slave_relay_log_info`

```
master_info_repository     = TABLE
relay_log_info_repository  = TABLE
relay_log_recovery         = 1
```

- Noch etwas buggy...

Multi-threaded Slaves

- Alle gegen einen: SQL Thread



Parallele Worker Threads

- In 5.6: Parallele Event Ausführung (multi-Threaded Slave)
  - Parallelisieren pro Schema
  - Gut für Hoster mit Traffic auf vielen Schemata!

```
slave_parallel_workers = <n>
```

- Slave-Lag → Galera Replikation!

Delayed Replikation

- Zeitverzögerte Replikation

  Früher `mk-slave-delay` (Maatkit)

```
CHANGE MASTER TO MASTER_DELAY = n;
```

- Slave kann auf ein Netzwerk Interface gebunden werden:

```
MASTER_BIND='eth1'
```

Remote Binary Log Shipping

- Remote Binary Log Shipping:

```
mysqlbinlog --read-from-remote-server
--raw bin-log.000001 > bin-
log.000001.dup
```

- Timestamp in `SHOW SLAVE STATUS`:

```
SHOW SLAVE STATUS\G
...
      Last_IO_Error_Timestamp: 120130 16:59:12
      Last_SQL_Error_Timestamp:
```

Q & A

Fragen ?

Diskussion?

Wir haben Zeit für ein persönliches Gespräch...

- **FromDual bietet neutral und unabhängig:**
  - **Beratung**
  - **Remote-DBA**
  - **Support für MySQL, Galera, Percona Server und MariaDB**
  - **Schulung**

www.fromdual.com/presentations