MySQL Replikation für Einsteiger

FrOSCon 2011, St. Augustin

Oli Sennhauser

Senior MySQL Consultant at FromDual GmbH

oli.sennhauser@fromdual.com

Über FromDual GmbH

- FromDual bietet neutral und unabhängig:
  - Beratung für MySQL (vor Ort und remote)
  - Remote-DBA Dienstleistungen / MySQL Betrieb
  - Premium Support (ex. MySQL Basic und Silber)
  - Schulung für MySQL
- Consulting Partner der Open Database Alliance (ODBA.org)
- Oracle Silber Partner (OPN)
- Mehr Informationen unter:

http://www.fromdual.com

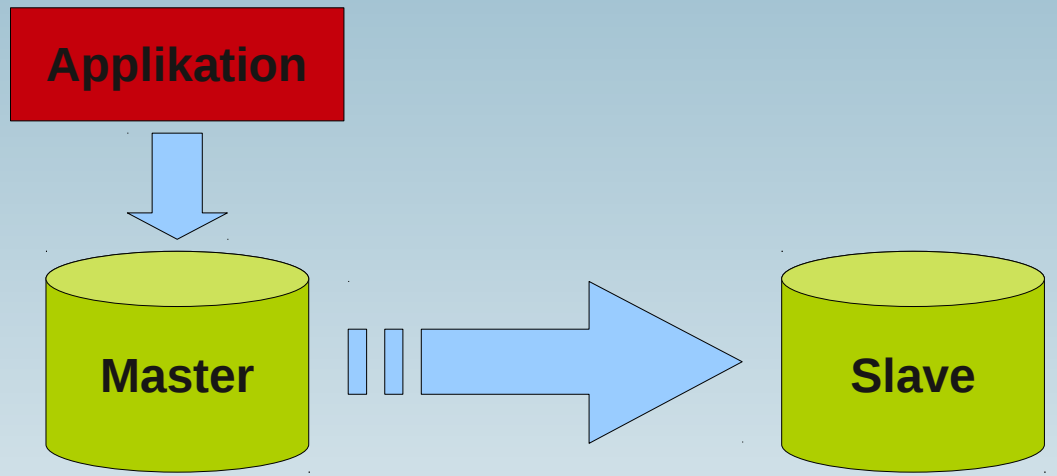
Inhalt

MySQL Replikation

- Was ist Replikation?
- Wie funktioniert MySQL Replikation?
- Wie wird Replikation aufgesetzt?
- Wann brauche ich MySQL Replikation?
- Neuerungen in 5.1, 5.5 und 5.6
- Varianten der Replikation

Was ist Replikation?

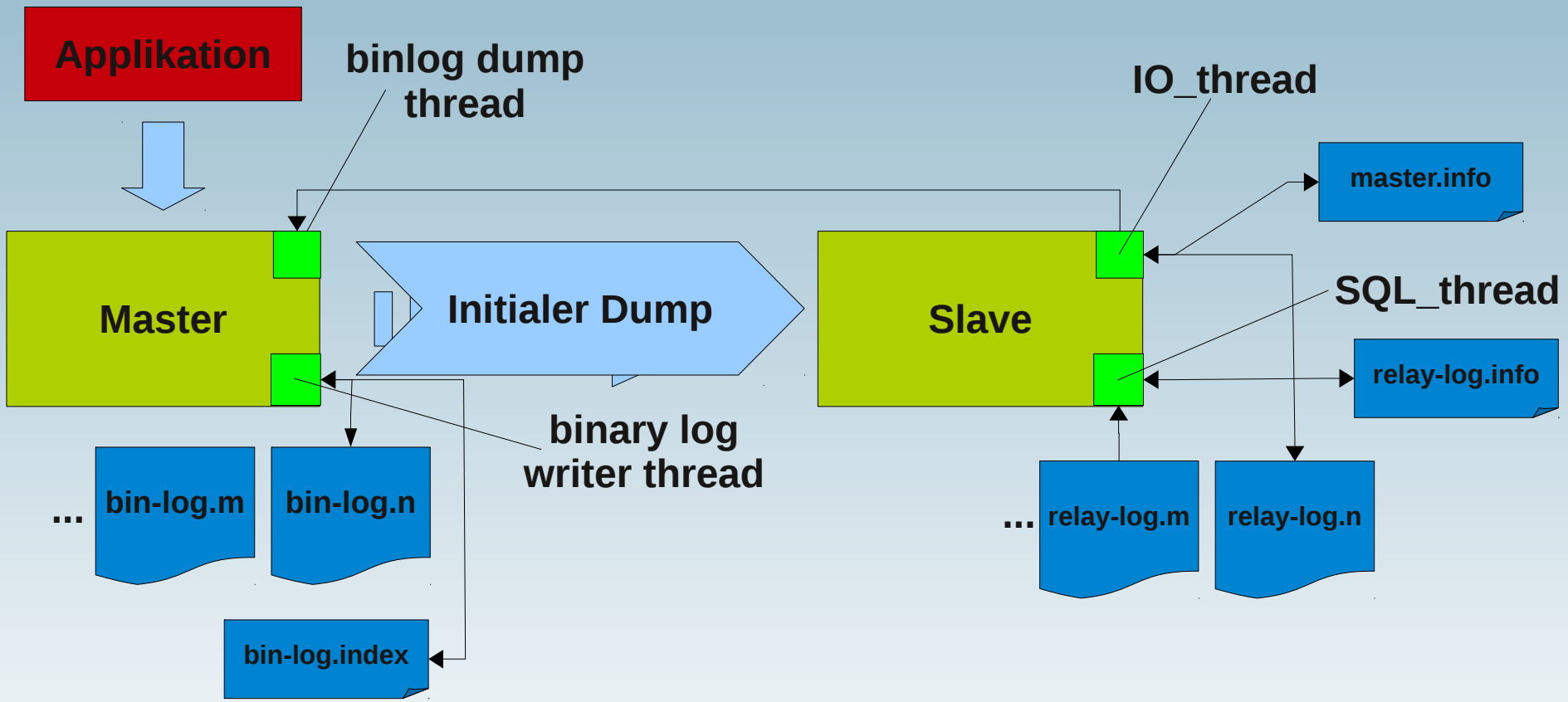
- **Daten von einer Datenbank auf eine oder mehrere andere Datenbanken weiterreichen:**



- **Daten = `UPDATE`, `INSERT`, `DELETE`, ... (= DML Statements) oder (binäre) „Events“**

MySQL Replikation

- **Wie funktioniert die MySQL Replikation?**



Vorbereiten des Masters

- Auf dem Master im `my.cnf`:
  - Binary log einschalten:

    ```
    log-bin = bin-log
    ```

  - Server ID setzen:

    ```
    server_id = 42
    ```

    - Muss im Replikations-Setup Unique sein!
- Server neu starten (downtime!)

User anlegen und Dump ziehen

- Immer noch auf dem Master:
- Replikations-User anlegen:

```
CREATE USER 'replication'@'192.168.1.%'
IDENTIFIED BY 'secret';

GRANT REPLICATION SLAVE ON *.*
TO 'replication'@'192.168.1.%';
```

- Initialen konsistenten! Dump ziehen:

```
mysqldump --all-databases --single-
transaction (--lock-all-tables)
--master-data > full_dump.sql
```

Aufsetzen des Slaves

- Neue Datenbank erstellen (2. Maschine)

```
./scripts/mysql_install_db \
--datadir=/var/lib/mysql
```

- Andere Server ID setzen:
  - `server_id = 43`
- Slave auf seinen Master ansetzen:
  - `CHANGE MASTER TO master_host='192.168.1.42' , master_port=3306, master_user='replication' , master_password='secret';`
- Initialen konsistenten Dump einspielen:

```
mysql -u root < full_dump.sql
```

Kontrolle und starten

- **Kontrolle:**

```
SHOW SLAVE STATUS\G
```

```
      Slave_IO_State:
         Master_Host: 192.168.1.42
         Master_User: replication
         Master_Port: 3306
       Master_Log_File: bin-log.000001
   Read_Master_Log_Pos: 51819
        Relay_Log_File: relay-bin.000001
         Relay_Log_Pos: 4
      Slave_IO_Running: No
     Slave_SQL_Running: No
```

- **Starten des Slaves:**

```
SLAVE START;
```

Probleme beim Aufsetzen

- MySQL Replikation funktioniert grundsätzlich gut!
- Gründe warum Probleme auftreten:
  - Nicht sauberes befolgen der Anleitungen!
  - Nicht konsistentes Backup
    - ohne `--single-transaction` / `--lock-all-tables` :-(
    - Jedes Schema einzeln (`--all-databases`) :-(
- MySQL Dokumentation: How to Set Up Replication

  http://dev.mysql.com/doc/refman/5.1/en/replication-howto.html

Probleme beim Betrieb

- Slave wird nicht überwacht
- Binary logs werden nicht aufgeräumt

```
expire_logs_days = n
PURGE BINARY LOGS TO 'bin-log.000013';
```

- Fummeln auf dem Slave
- Master/Slave Lag (hinterher hinken)
- Master/Slave Drift (auseinander laufen)

  → Wenn Master und Slave auseinander laufen MUSS der Slave neu aufgesetzt werden

- Filtern auf dem Master ist meist keine gute Idee!

Master/Slave Drift

- Master und Slave können auseinander laufen weil:
  - Auf dem Slave geschrieben wird (rumfummeln!)
  - `sql_bin_log = 0`
  - `sql_slave_skip_counter = 1`
  - `slave_skip_errors=...`
  - Nicht-deterministische Abfragen/Funktionen → viel besser mit Row Based Replikation (RBR)
  - Filtern auf Master oder Slave
  - Temporäre Tabellen mit SBR
- Unterschiede können gefunden werden mit Tools aus dem Maatkit:

  `mk-table-checksum`
- und gefixed werden mit:

  `mk-table-sync`
- Ansonsten → Slave neu aufsetzen!!!

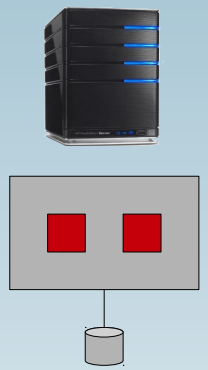
Wann brauche ich die MySQL Replikation?

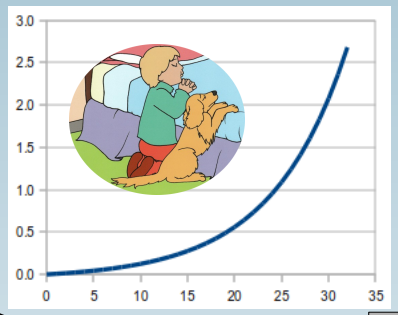
- Es gibt verschiedene Szenarien:
  - Scale-Out Lösungen
  - Backup Slave
  - Datenanalyse, Reporting
  - Hochverfügbarkeit (HA)
  - Zeitversetzte Datenstände vorhalten

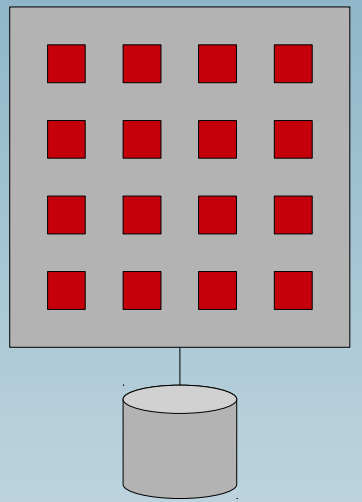
MySQL Scale-Out vs Scale-Up

Scale-Up

- Kosten
- MySQL Design
- Phsikalische Flaschenhälse

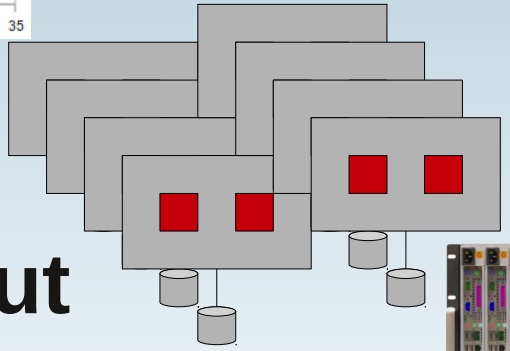




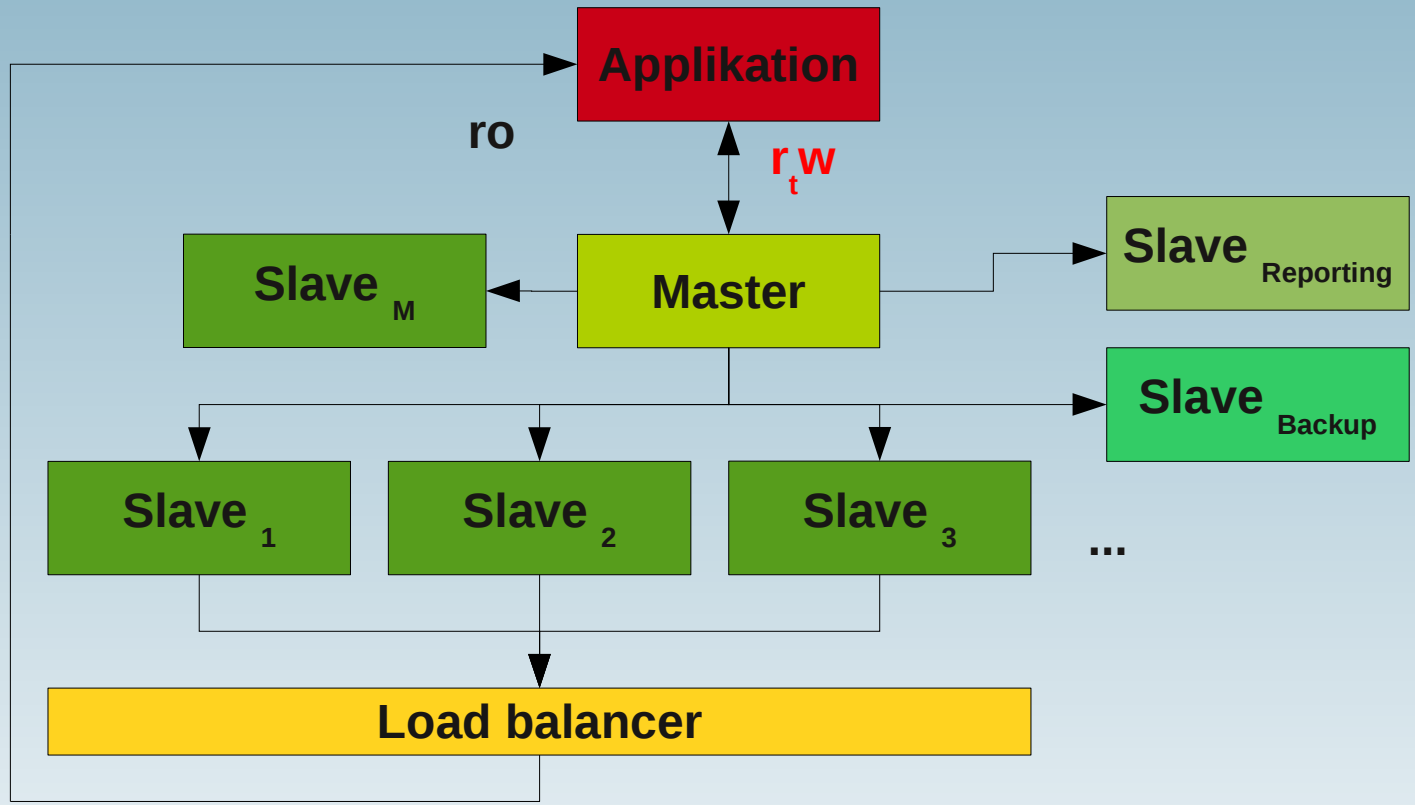


- „Relaxation of Constraints“

Scale-Out



Der MySQL Scale-Out Ansatz



Neuerungen in MySQL 5.1, 5.5 und 5.6

- Row-Based Replikation (5.1)
- Semi-Synchrone Replikation (5.5)
- Row Image Control (5.6)
- Crash Safe Binary Logs (5.6)
- Remote binary log shipping (5.6)
- ...

Row-Based Replikation (5.1)

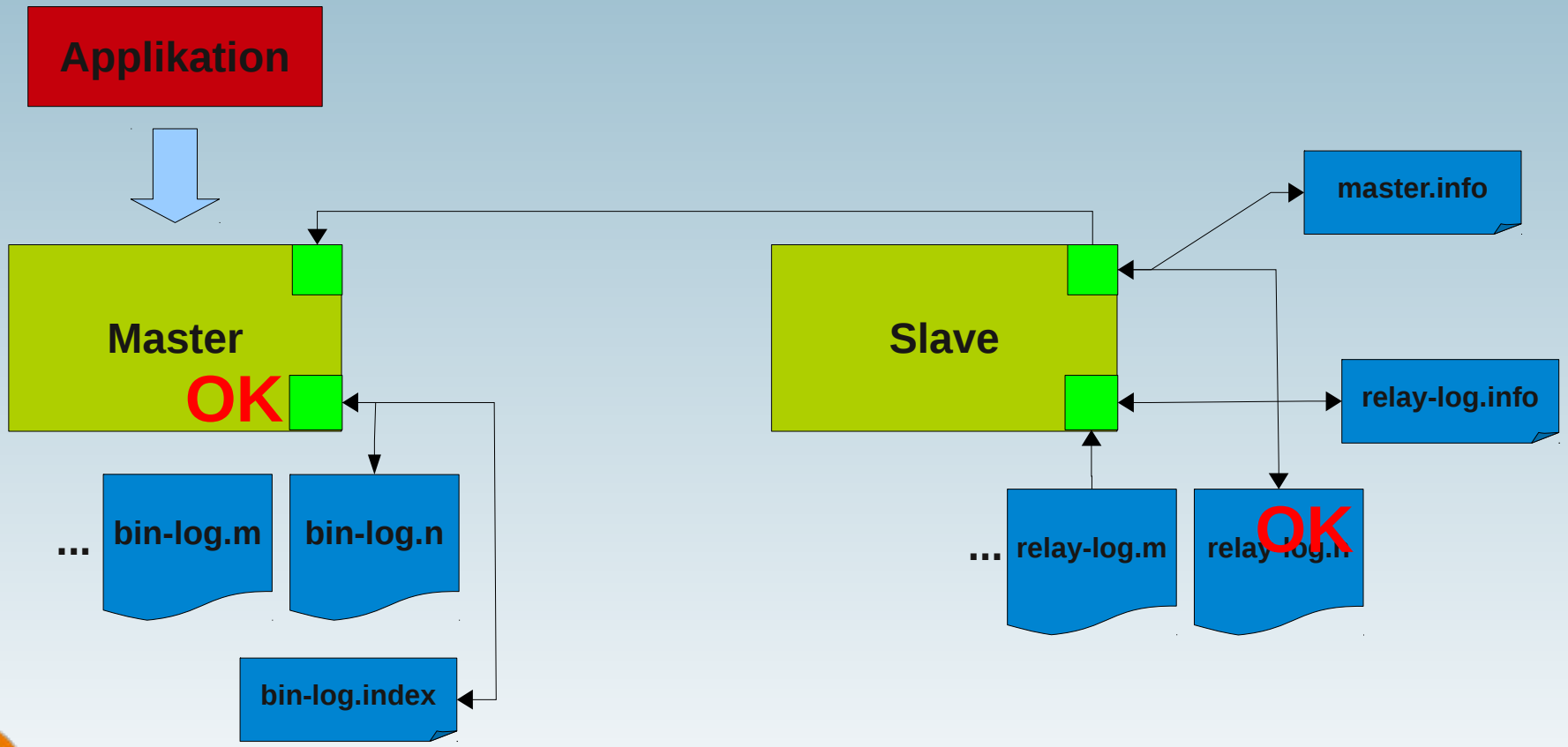
- Alt (<= 5.0) Statement-Based Replikation (SBR)
  - → Statements werden übermittelt
  - Problem: nicht-deterministische Abfragen/Funktionen
    → inkonsistente Daten zwischen Master und Slave
- Neu (>= 5.1) Row-Based Replikation (RBR)
  - → Events / Trx werden übermittelt
  - `binlog_format = {ROW | MIXED | STATEMENT}`
  - RBR ist die sicherste Art der Replikation!
  - Wie sehe ich trotzdem was passiert?
    `mysqlbinlog --base64-output=DECODE-ROWS --verbose bin-log.000069`
  - Mehr binlog Traffic, unterschiedliche Performance → Testen!

Semi-Synchrone Replikation (5.5)

- Default asynchrone Replikation
  - 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

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



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;
```

Replikation in MySQL 5.6

- Row Image Control
  - `binlog_row_image = {full | minimal | noblob}`
- Crash safe Binary Logs
  - Vollständige Events/Trx werden geloggt
  - Event-Länge + CRC32 Checksumme
- Slave: `master.info` und `relay-log.info` zusätzlich in (MyISAM) Tabellen
  - `slave_master_info` + `slave_relay_log_info`
  - Umwandeln in InnoDB?

Replikation in MySQL 5.6

- Remote Binary Log Shipping:

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

- Delayed Replication

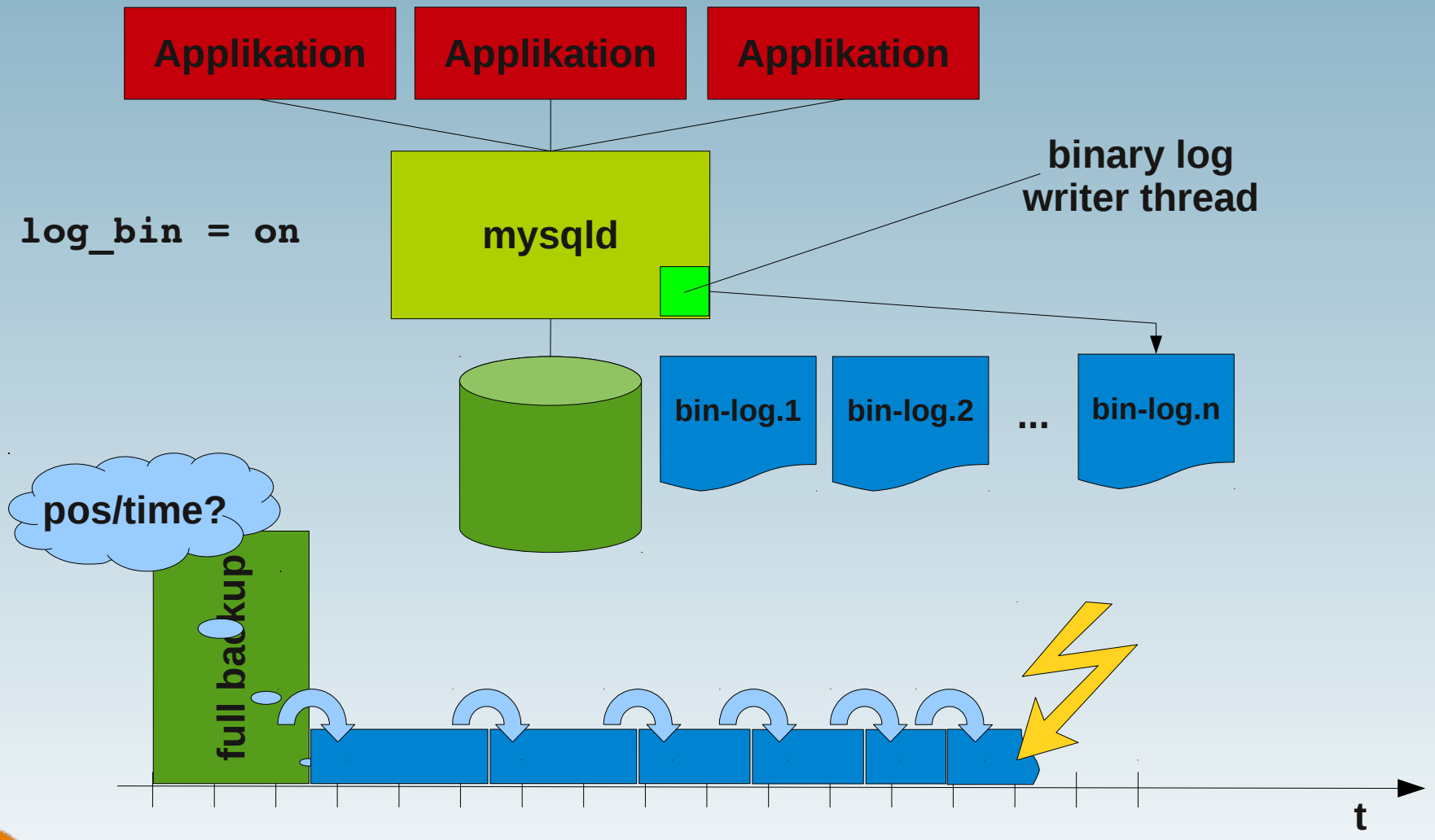
```
CHANGE MASTER TO MASTER_DELAY=n;
```

  - Früher `mk-slave-delay` (Maatkit)
- Neue Variablen:
  - `log_bin_basename` und `log_bin`
- Aus den Oracle Labs:
  - Multi-threaded Slave / Parallel Slave
  - Multi-source Replikation?

Varianten der Replikation

- **Bianary Log für Point-in-Time-Recovery**
- **Tungsten Replication Cluster**
- **Synchone Replikation mit Galera**

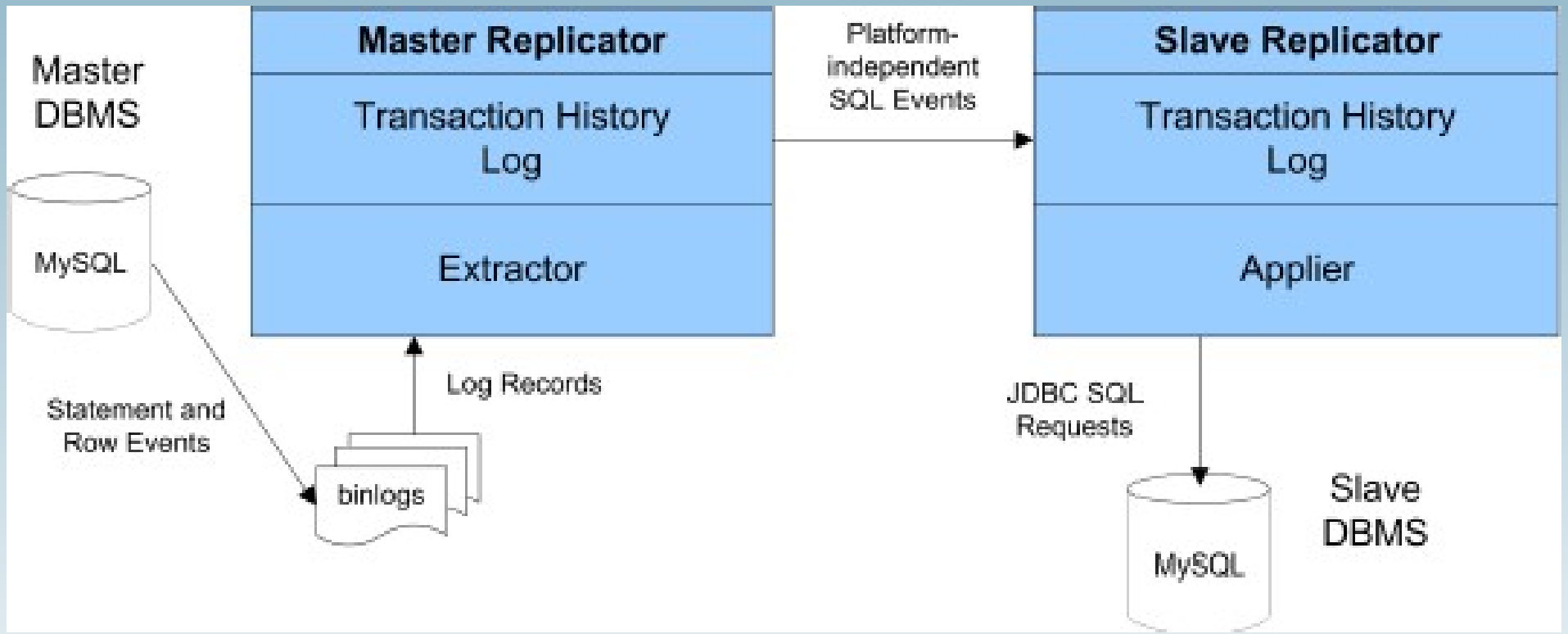
Point-in-Time-Recovery (PITR)



Tungsten Replication Cluster

- Continuent (www.continuent.com)
- Master/Slave Replikation (asynchron)
  - Basiert auf MySQL Binary Logs!
- Verfügbarkeit / Performance / Integrität
- Für MySQL und PostgreSQL!
- Benötigt Java und Ruby
  - Führt eine “globale Transaktions-ID” ein
- Multi-Source Replikation möglich!

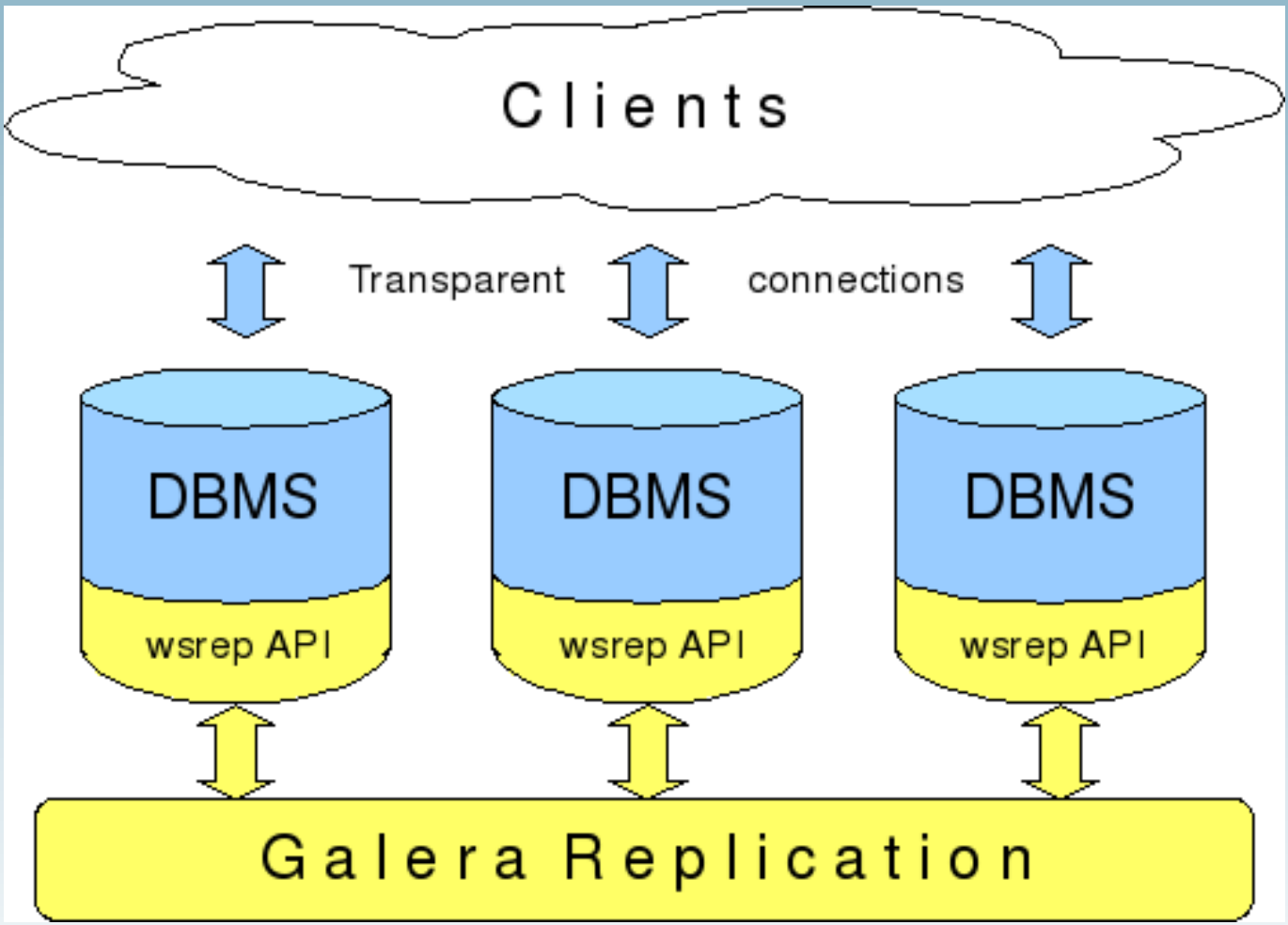
MySQL Replikationsarchitektur



Synchrone Replikation mittels Galera

- Codership (www.codership.com)
- Synchrone Replikation
  - “Certification Based Replication”, kein 2PC!
- Verfügbarkeit / Skalierbarkeit / transparent
- Änderungen am MySQL Code notwendig!
  - MySQL muss dazu gepatched und neu kompiliert werden...
  - Codership liefert fertige Binaries!

Galera Replication



Q & A

Fragen ?

Diskussion?

Wir haben noch Zeit für persönliche und indviduelle Beratungen...