

MySQL Replikation, Scale-Out, Master-Master Replikation, Backup

**DOAG Regioaltreffen, München
23. März 2011**

Oli Sennhauser

Senior MySQL Consultant, FromDual GmbH

oli.sennhauser@fromdual.com



Inhalt

MySQL Replikation

- › **Scale-Out**
- › **MySQL Replikation**
- › **Master-Master Replikation**
- › **Binary Log Formate**
- › **Semi-Synchrone Replikation**
- › **Backup Methoden**
- › **Restore**
- › **SE Replikation mit PBXT**



Über FromDual GmbH

- **Wir bieten an:**
 - Neutrale und Hersteller unabhängige Beratung für MySQL
 - Support für MySQL (7 x 24)
 - Remote-DBA / MySQL Betrieb (wir betreiben Ihre MySQL DB!)
 - Schulung und Workshops (DBA, Performance Tuning, Scale-Out, High Availability, MySQL Cluster)
- **Wir sind:**
 - Consulting Partner der Open Database Alliance (ODBA.org)
 - Oracle Silber Partner (OPN)

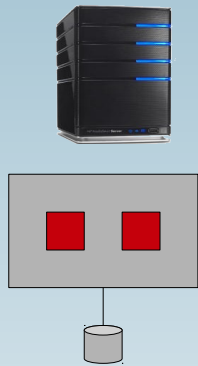


<http://www.fromdual.com>

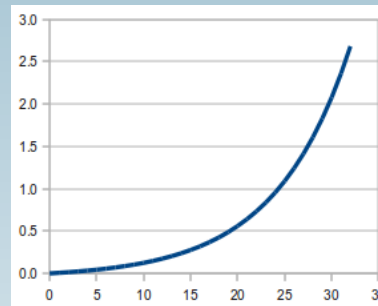
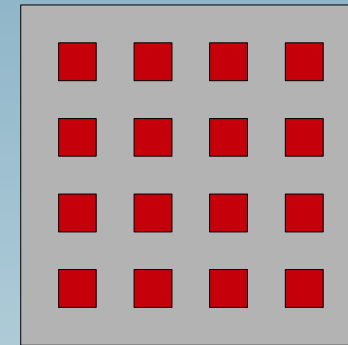


Zuerst gab's da mal ein Problem:

- Kosten
- MySQL Design
- Physikalische Flaschenhalse
- „Relaxation of Constraints“

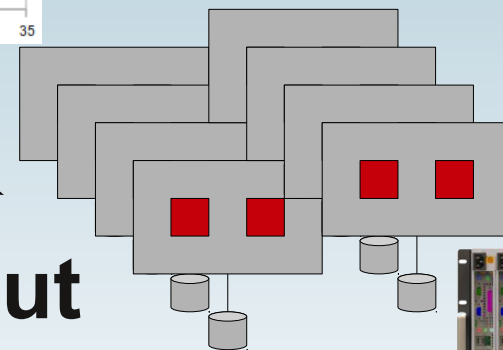


Scale-Up

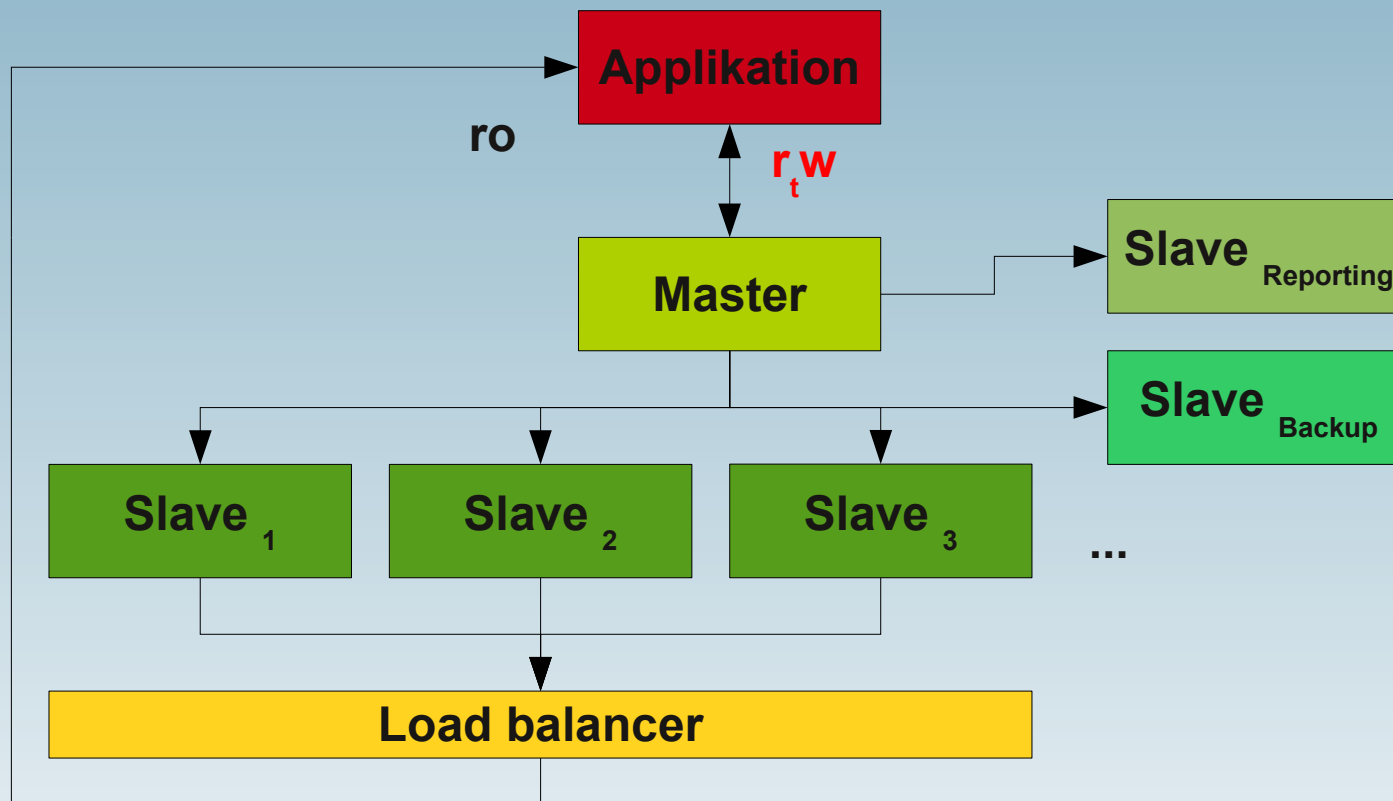


- Web-Applikationen typischerweise:
 $r \gg w$

Scale-Out



Der MySQL Scale-Out Ansatz

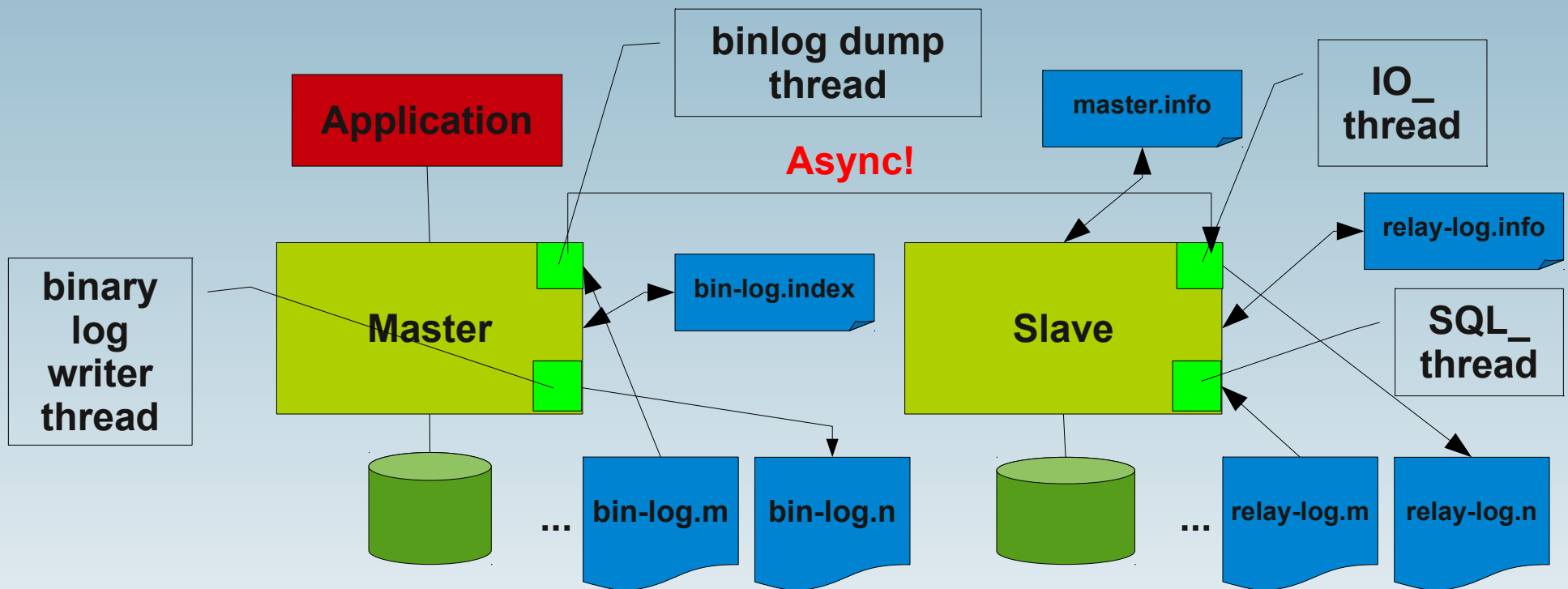


Web-Applikationen typischerweise:

$r \gg w$



Master – Slave Replikation



Erstellen eines Masters

- Binary Log einschalten und Server ID setzen (erfordert Neustart):

```
# my.cnf
[mysqld]
log_bin      = binary_log
server_id    = 42
```

- User mit REPLICATION SLAVE Privileg anlegen:

```
CREATE USER replication@'%' IDENTIFIED BY 'secret';
GRANT REPLICATION SLAVE ON *.* TO replication@'%';
```

- Konsistentes Backup vom Master erstellen:

```
mysqldump --all-databases {--single-transaction | --lock-
all-tables} --master-data > full_dump.sql
```

- MySQL Dokumentation: How to Set Up Replication

<http://dev.mysql.com/doc/refman/5.5/en/replication-howto.html>

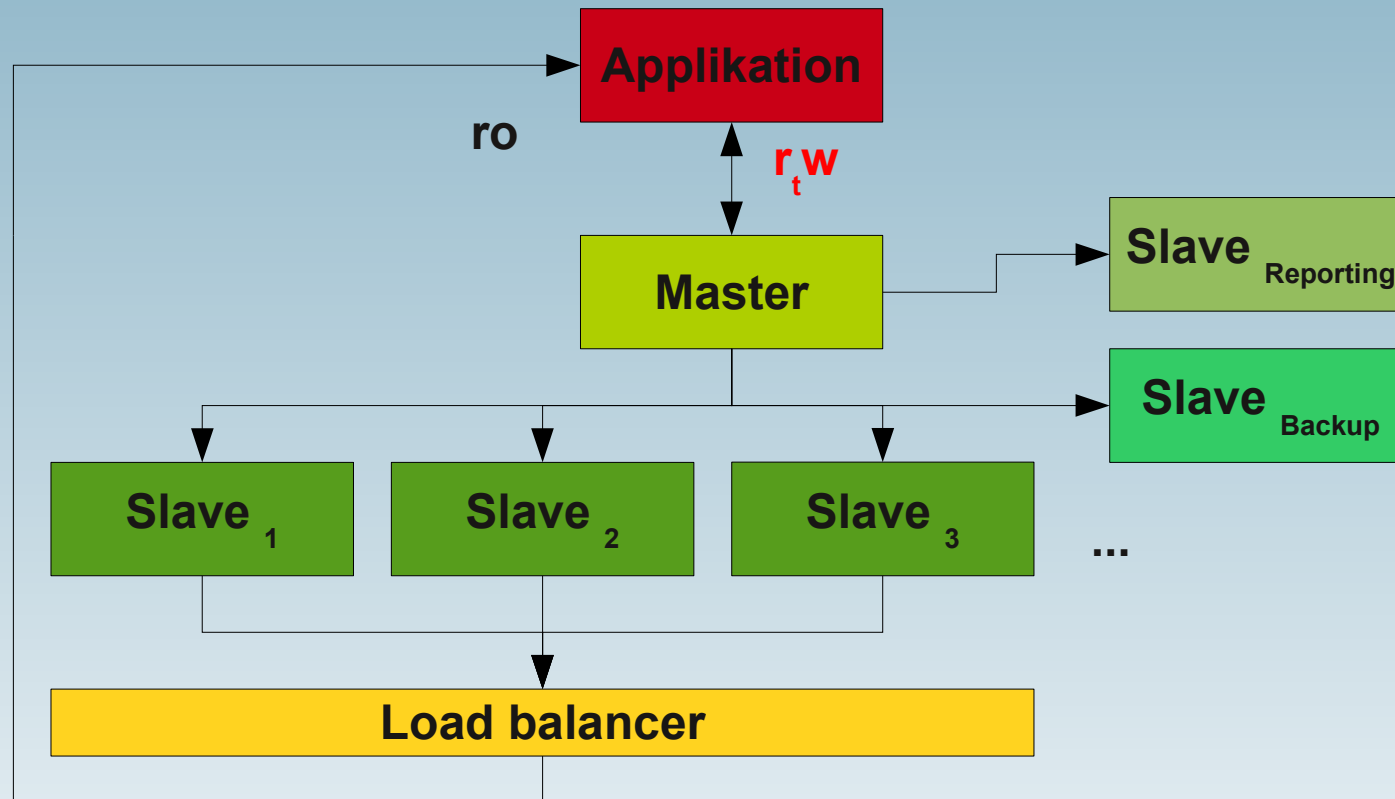


Erstellen eines Slaves

- `Server_id` in `my.cnf` setzen
- Dem Slave zeigen, wo sein Master sitzt:
 - `CHANGE MASTER TO master_host='masterserver', master_port=3306, master_user='replication', master_password='secret'`
- Einspielen des konsistenten Backups vom Master:
`mysql -u root < full_dump.sql`
- Überprüfen ob auf dem Slave alles i.O. ist:
`SHOW SLAVE STATUS\G`
- Slave starten:
`START SLAVE;`



Für was kann man Slaves alles brauchen?

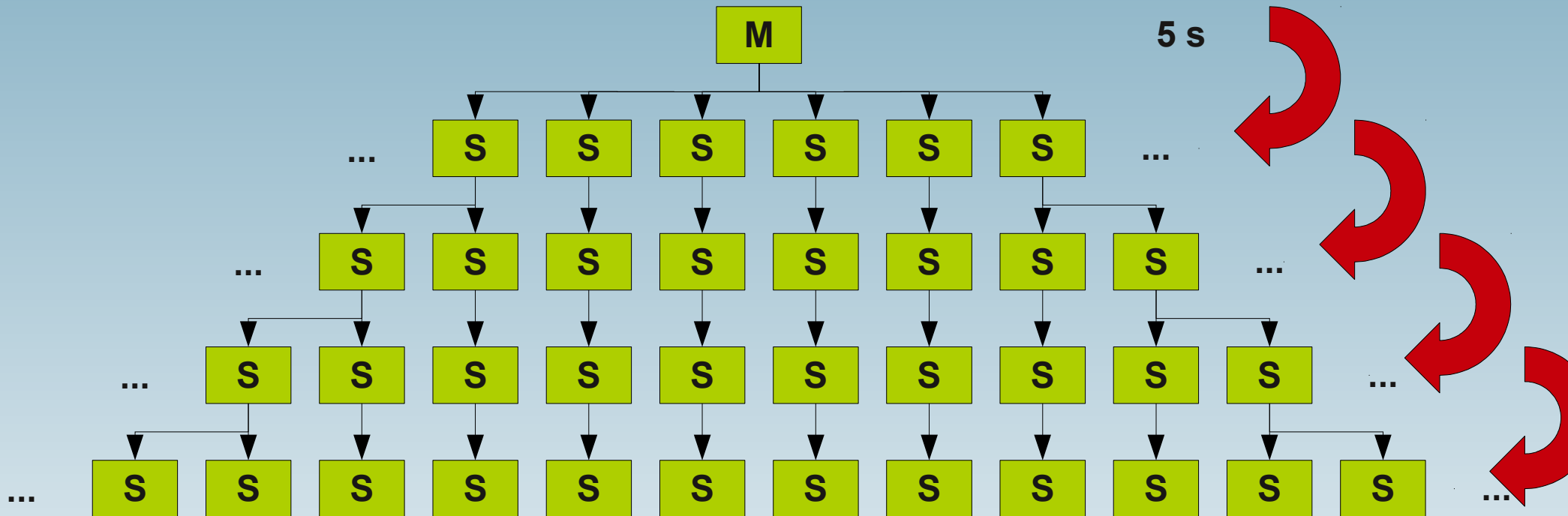


Web-Applikationen typischerweise:

$r \gg w$



Massives Scale-Out



- Jetzt können sie auch erahnen wozu die **BLACKHOLE SE** gebraucht wird...

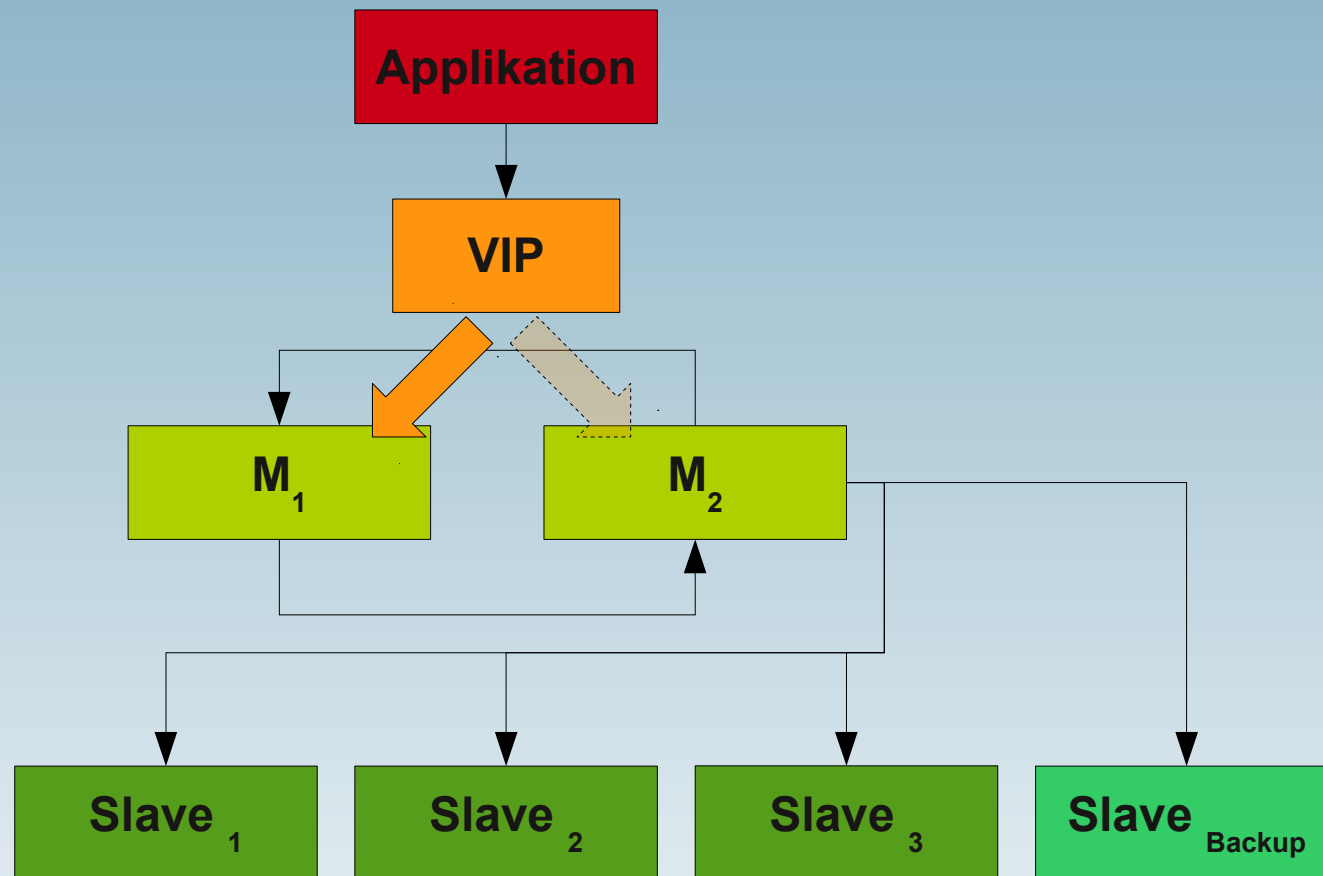


Tool zur Verwaltung solcher Set-up's

- **Problem: Wenn Master kaputt geht, müssen alle Slaves neu gebaut werden...**
- **Aber es gibt Tricks:**
- **Tool von Yandex.ru (Petya Kohts):**
 - **mmmfm, multi-master mysql failover**
 - **<http://www.nigilist.ru/nit/mmmf/>**
 - **<http://cpan.uwinnipeg.ca/~kohts/mmmf>**



Master – Master Replikation



- **Vorsicht beim Schreiben auf beide Master**
- **Man erhält so NICHT mehr I/O-Durchsatz!**



Tool zur Verwaltung solcher Set-up's

- Ähnliches Problem wie bei Master-Slave Replikation mit zusätzlicher Komplexität durch zirkuläre Replikation
- Keine Konflikt-Detektion/Auflösung!
- Achtung: Wir sind asynchron!
- Tool:
 - Multi-Master Replication Manager for MySQL
 - <http://mysql-mmm.org/>
 - <https://launchpad.net/mysql-mmm>



Binary Log Formate

- **Bis MySQL 5.0 nur Statement Based Replikation (SBR)**
 - + **Weniger Traffic**
 - **Non-deterministic Queries!**
 - **Mehr Locking (grössere Locks)**

```
# at 786
#110321 20:55:40 server id 35154  end_log_pos 814          Intvar
SET INSERT_ID=3/*!*/;
# at 814
#110321 20:55:40 server id 35154  end_log_pos 944          Query   thread_id=3      exec_time=0      error_code=0
SET TIMESTAMP=1300737340/*!*/;
INSERT INTO test VALUES (NULL, 'Statement based replikation', NULL)
```



Binary Log Formate

- Ab MySQL 5.1 auch Row Based Replication (RBR)
 - + All Änderungen können nun repliziert werden.
 - + Weniger Locking / manchmal schneller
 - Binary Log ist weniger transparent / mehr Traffic
- Mixed (SBR default, wechselt auf RBR b. Bed.)

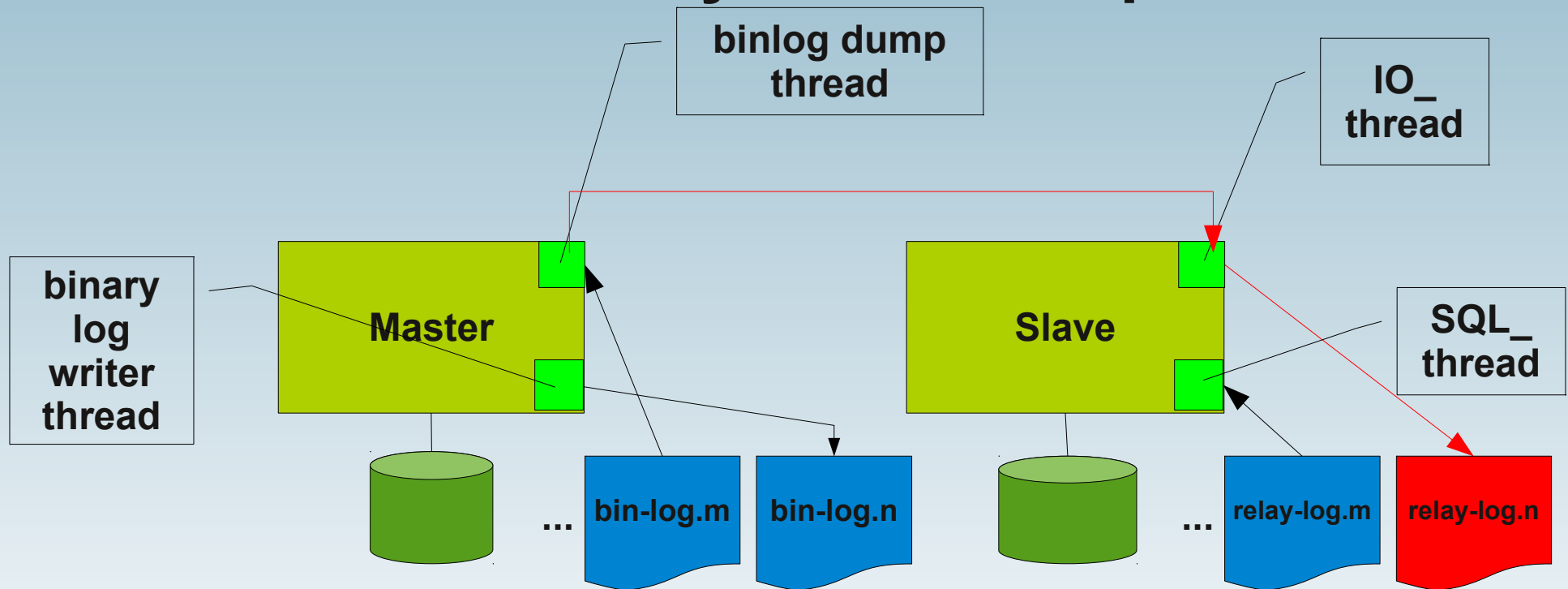
```
# at 844
#110321 20:57:36 server id 35154  end_log_pos 912      Query    thread_id=2      exec_time=0      error_code=0
SET TIMESTAMP=1300737456/*!*/;
BEGIN
/*!*/;
# at 912
# at 959
#110321 20:57:36 server id 35154  end_log_pos 959      Table_map: `test`.`test` mapped to number 15
#110321 20:57:36 server id 35154  end_log_pos 1019    Write_rows: table id 15 flags: STMT_END_F
BINLOG 'sK2HTRNSiQAALwAAAL8DAAAAAAAA8AAAAAAAEABHRlc3QABHRlc3QAawMPBwJAAAI=
sK2HTRdSiQAAPAAAAPsDAAAAAAAA8AAAAAAAEAA//4BwAAABVSb3cgYmFzZWQgcmlrYXRpb26w
RydN'/*!*/;
```

```
mysqlbinlog --base64-output=decode-rows --verbose bin-log.000002
```



Semi-Synchrone Replikation

- Bis 5.1 nur asynchrone Replikation!
- Ab 5.5 auch semi-synchrone Replikation:



Backup Methoden



Backup Methoden

- `mysqldump`
- **Filesystem Kopie / `mysqlhotcopy`**
- **Snapshot mit LVM / `btrfs`**
- **InnoDB Hot Backup (`ibbackup/xtrabackup`)**
- **Backup Replikation Slave**



mysqldump

- **Charakteristik**
 - **Logisches Backup, hot/on-line, lokal oder remote, konsistent**
- **Vorteile**
 - **Einfach, lokal oder remote, Standard-Backup für MySQL, konsistent, Strukturdump möglich**
- **Nachteile**
 - **Kann falsch gemacht werden, blockiert MyISAM Tabellen fürs Schreiben, nicht geeignet für sehr grosse Datenmengen, Restore-Zeiten!**



Filesystem Kopie / mysqlhotcopy

- **Charakteristik**
 - **Physisches Backup, hot, local**
- **Vorteile**
 - **Schneller Restore, konsistent**
- **Nachteile**
 - **Nur bedingt für InnoDB geeignet, blockiert Tabellen für Schreibzugriffe.**



Snapshot mit LVM / btrfs

- **Charakteristik**
 - **Physisches Backup, hot, lokal**
- **Vorteile**
 - **Sehr schnelle Backup Methode, ziemlich schnelles Restore, konsistent**
- **Nachteile**
 - **Benötigt root Rechte, etwas komplizierter und Hardware intensiver, möglicherweise gefährlich mit InnoDB?**

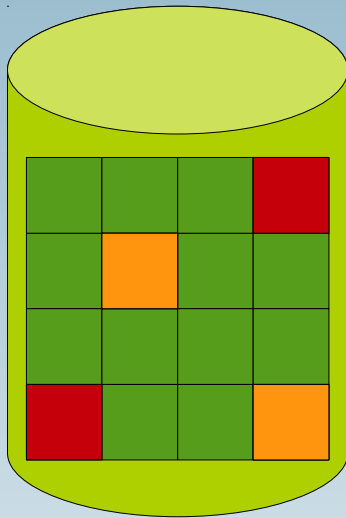


Wie wird ein LVM Snapshopt Backup gemacht?

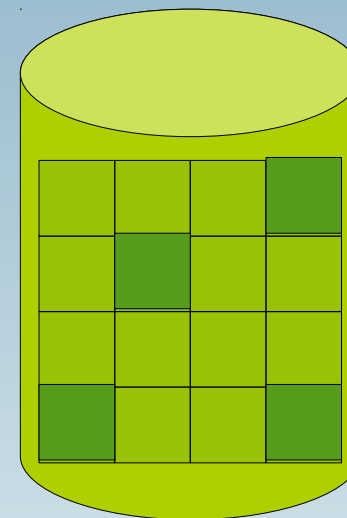
- Alle DB files müssen auf dem selben Logical Volume liegen (LV = Partition)
 - Locken der Datenbank
 - Erstellen eines Snapshots des Logical Volumes
 - Unlocken der Datenbank
 - Mounten des Snapshot Logical Volumes
 - InnoDB Recovery testen
 - Backup (tar, compress, tape) der Datenbank Files
 - Löschen des Snapshot Logical Volumes
- Etwas kompliziert: mylvmsnapshot



Wie funktionieren LVM Snapshots?



LVM device



LVM snapshot device

copy-on-write

www.fromdual.com



InnoDB Hot backup (ibbackup, xtrabackup)

- **Charakteristik**
 - **Physisches Backup, hot/on-line, lokal, konsistent, inkremental**
- **Vorteile**
 - **Schnelles Backup/Restore für InnoDB**
- **Nachteile**
 - **Löst die MyISAM Probleme nicht**



Backup mit OEB/XtraBackup

- Volles Backup erstellen:

```
xtrabackup --backup --target-dir=...
```

- Prepare (~recovery):

- `xtrabackup --prepare --target-dir=...`

- Restore?

- Einfach MySQL auf den Files wieder starten...



Backup Replikation Slave

- **Charakteristik**
 - Logisches oder physisches backup, hot/on-line, konsistent
- **Vorteile**
 - Betrifft Master überhaupt nicht. „Beste“ Methode aus Sicht des Masters.
- **Nachteile**
 - Braucht zusätzliche Hardware (oder zumindest Ressourcen)
 - Sicherstellen, dass Slave nicht driftet.
 - Achtung: Wir erstellen ein Backup auf dem Slave: != Master



Restore / Recovery

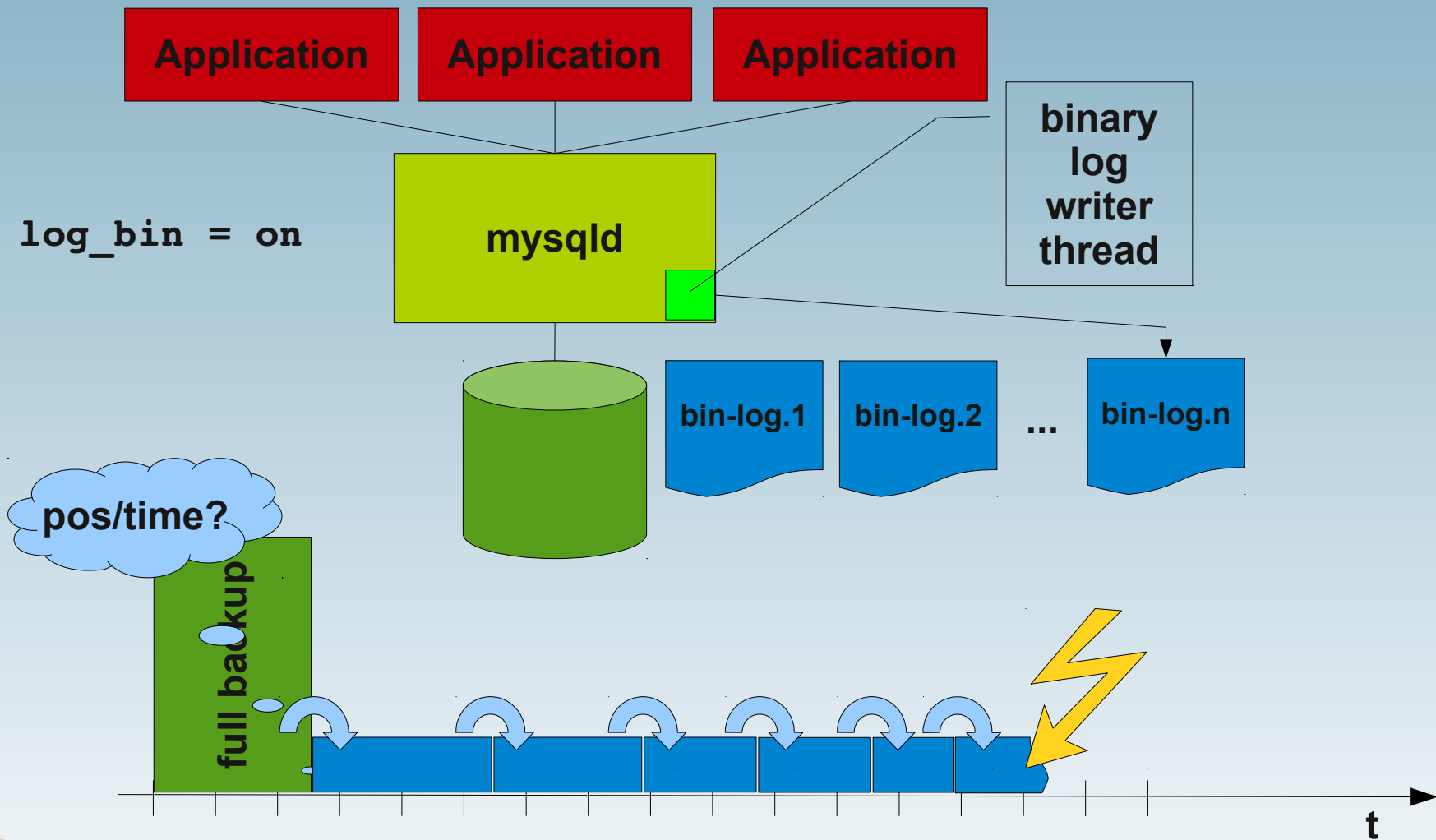


Restore / Recovery

- **Restore / Recovery besteht aus bis zu 3 verschiedenen Schritten:**
 - **Restore der Daten oder der Tabellen oder den Files**
 - **Auto-recovery der transaktionalen SE (InnoDB, PBXT, NDB)**
 - **Point-in-Time-Recovery (PITR)**
- **Schritt 1 und 2 sind klar?**
- **Was ist PITR?**



Point-in-Time-Recovery (PITR)

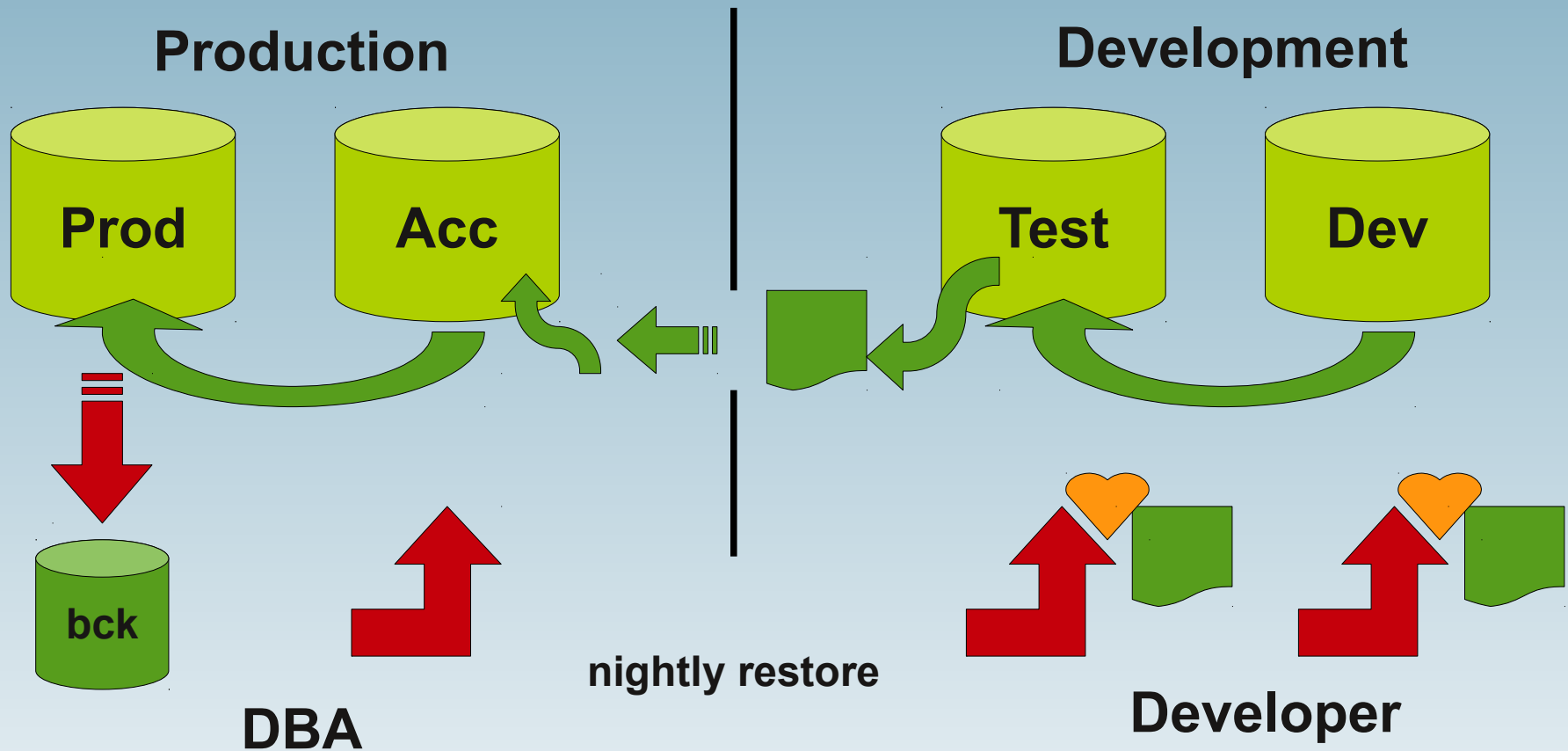


Warum ein Restore getestet werden sollte?

- **Restore können sehr sehr lange dauern.....**
 - **Überprüfen Sie Ihre MTTR!**
- **Warum dauert der Restore eines logischen Backups so lange?**
 - **Restore Tabelle, Create Index → random I/O**
 - **So lange Ihr Index in den Cache/Buffer passt ist es OK, aber wehe wenn nicht...!**
 - **Man kann da etwas drum herum schummeln mit Fast Index Creation im InnoDB Plug-in oder MySQL 5.5...**



Justify your restore



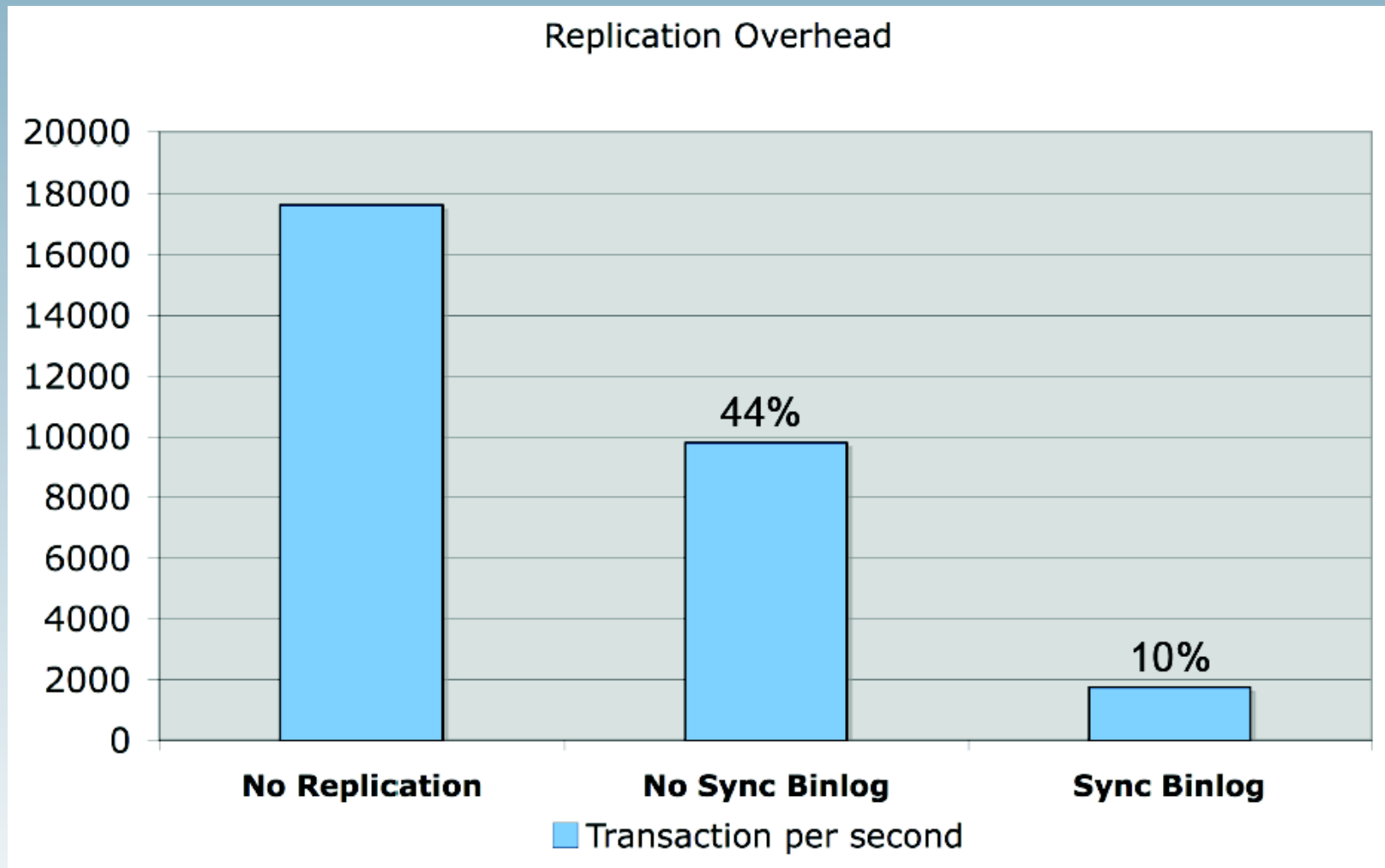
Storage Engine Replikation mit PBXT

- **MySQL Replikation ist nicht sehr schnell**
- **Man kann die Replikation leicht falsch aufsetzen / Fehler machen.**
- **Warum so kompliziert über mind. 3 Threads und 2 Files?**

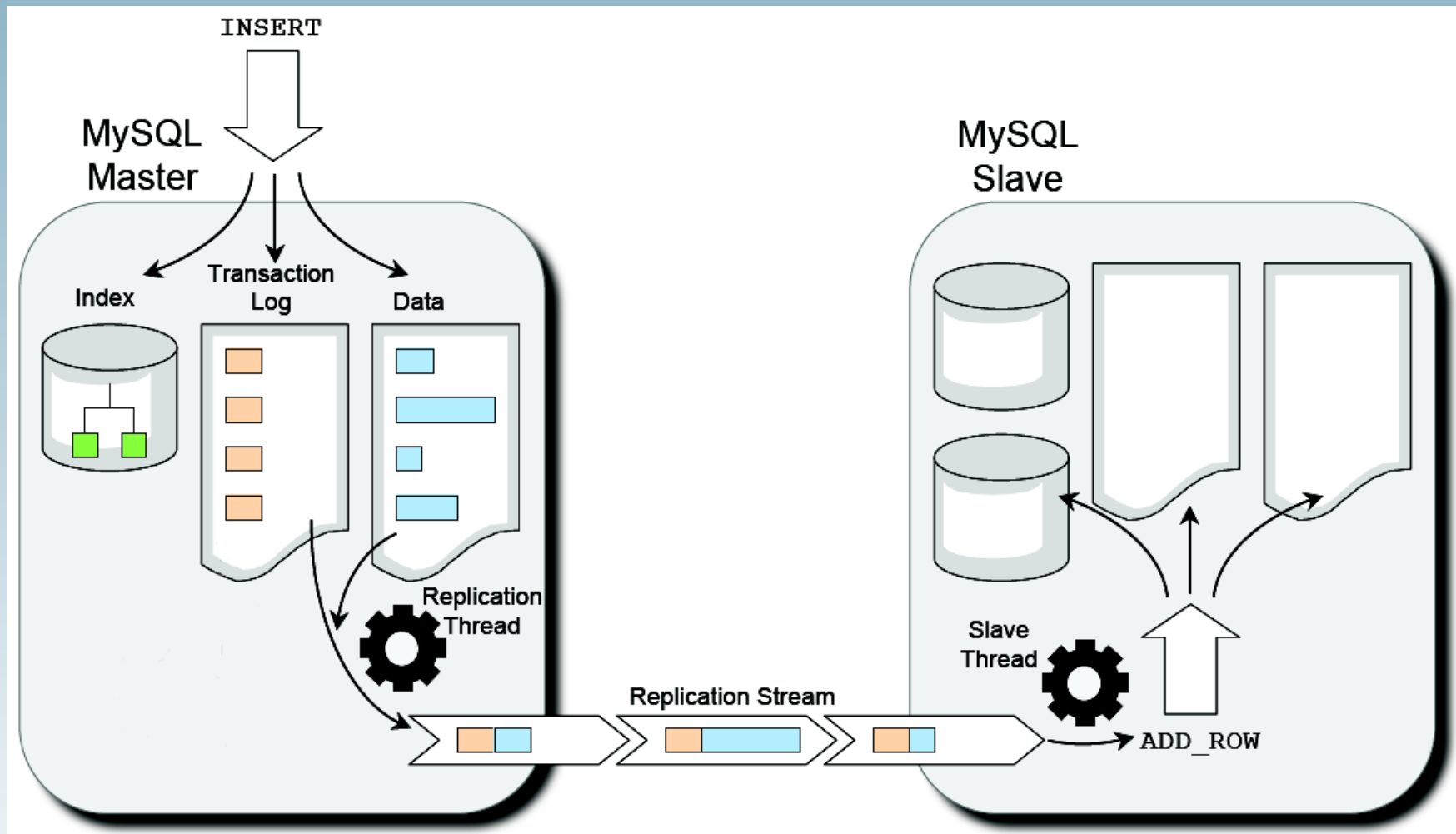
→ **Die PBXT SE v2.0 implementiert Replikation auf Storage Engine Ebene:**



Overhead der MySQL Replikation



Architektur der PBXT Replikation



Overhead der PBXT Replikation

