MySQL-Datenbanken sichern – Teil 2: Master-Slave-Betrieb

Im Schatten des Meisters

Das Live-Backup einer MySQL-Datenbank erlaubt kurze Ausfallzeiten bei aktuellem Datenbestand. Läuft MySQL im Master- und im Slave-Modus, sorgt die Datenbank auf dem zweiten Rechner selbst dafür, dass sie auf dem neuesten Stand ist. Mit mehreren Slaves lässt sich zudem eine einfache Lastverteilung realisieren. Thomas Wölfer

Wer die MySQL-Datenbank in einem produktiven Umfeld betreibt, muss für hohe Ausfallsicherheit sorgen. Dazu reicht es nicht, lediglich ein aktuelles Backup auf Band oder CD vorzuhalten. Besser ist eine Konstruktion aus zwei Rechnern, die als Master und Slave zusammenarbeiten. Der Slave erhält stets ein Live-Update der Datenbank und kann beim Ausfall des Masters schnell übernehmen. Zwar ist die Konfiguration dieses Modus relativ einfach, doch ist zu beachten, dass die einzelnen Schritte in einer bestimmten Reihenfolge durchzuführen sind.

Lastverteilung inklusive

Mit einem Slave lässt sich auch die Last in einem Netzwerk praktisch verteilen. Kommt MySQL auf einer Website zum Einsatz, die mit einer Volltextsuche ausgestattet ist, blockieren die Suchvorgänge sehr schnell den Datenbankserver. Übernimmt hingegen der Slave die anstrengende Suche nach Texten, kann der Master-Rechner mit voller Leistung seine eigentlichen Pflichten erfüllen. Im schlimmsten Fall blockieren sich die Suchenden lediglich selbst.

Ein weiterer Vorteil ist außerdem, dass eine Master-Slave-Konfiguration nur wenig Wartung verlangt und somit auch kaum die stets knappen Ressourcen der Administratoren in Anspruch nimmt.

Mehr Sicherheit durch Replikation

Der Master- und Slave-Modus ist seit der MySQL-Version 3.23.15 implementiert. Wegen diverser Bugs empfehlen die Macher jedoch mindestens die Version 3.23.30 einzusetzen. Der Modus erlaubt eine Topologie aufzubauen, in der sich eine MySQL-Datenbank auf einem primären Server auf weitere MySQL-Server abbilden lässt. Diese Replikation funktioniert automatisch, was den Modus mehrfach interessant macht. Das Live-Backup auf einem zweiten Rechner dient als Failover, das der Administrator jedoch manuell anfahren muss. Auch Cluster profitieren, da sich durch das Setup die Last geschickt zwischen den einzelnen Nodes aufteilen lässt.

Das Replikationsmodell bei MySQL unterstützt die Ein-Weg-Replikation: Ein Server fungiert als Master und beliebig viele andere Rechner agieren als Slave. Ein Slave kann seinerseits auch wieder Master sein. Soll die Datenbank also besonders ausfallsicher sein, dient eine Kette von Servern dazu, das Master-Slave-Modell zu übernehmen und somit zu jeder Zeit die aktuellen Daten online vorzuhalten.

Als einziger Sicherheitsmechanismus reicht ein Slave-Rechner nicht aus. Zwar ist ein Slave das perfekte Backup, wenn der Master aufgrund eines Hardwarefehlers ausfällt. Allerdings können Softwarefehler Desaster auslösen. Da der Slave alle Updates des Masters live mitführt, lösen falsche SQL-Instruktionen entsprechende Katastrophen aus. Ein irrtümlich abgesetztes »delete from members;« auf dem Master führt dazu, dass die Mitgliedertabelle auch auf dem Slave sofort verschwunden ist. Ein konventionelles Backup ist also auch im Master-Slave-Betrieb Pflicht.

Hinter den Kulissen

Der Master-Server protokolliert alle Updates der Datenbank im so genannten Binary-Log. Nachdem sich der Slave mit dem Master verbunden hat, teilt er seine eigene Position in diesem Log mit. Fortan informiert der Master den Slave über Updates. Der Slave holt sich die aktuellen Datensätze ab und hostet somit live eine identische Datenbank. Sind Master und Slave auf dem gleichen Stand, wartet der Slave, bis ihn der Master über ein Update informiert. Wichtig ist dabei vor allem, dass die echten Updates nur der Master-Rechner bekommt. Sonst laufen die Datenbestände der beteiligten Rechner auseinander.

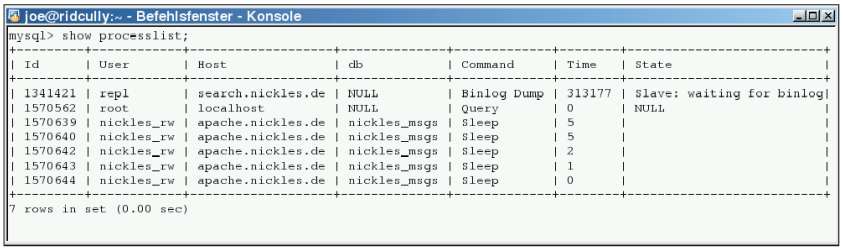

Abbildung 1: Die Prozessliste auf dem Master gibt über den Status der Slaves bereitwillig Auskunft.

Für die Replikation ist es zunächst notwendig, dass der Master das Binary-Log führt. Die Protokolldatei enthält nicht nur alle Updates der primären Datenbank, das Log ist zudem für den Slave wichtig. Er benötigt als Einstieg den gleichen Datensatz wie der Server, als dieser das Binary-Log in Betrieb genommen hat. Dieser Datensatz muss irgendwie auf den Slave wandern.

Dazu hält der Admin den Server an und kopiert alle Daten in die Slave-Datenbank. Welche Optionen beim Kopieren zwischen zwei MySQL-Servern bereitstehen und wie diese anzuwenden sind, erläutert der erste Teil dieser Reihe [1]. Ist die Kopie eingespielt, aktiviert man das Binary-Log und startet den Master-Server. Wann anschließend der Slave startet, ist egal: Sofort nach dem Start beginnt er unbeirrt die Daten von seinem Master abzuholen. Startschuss ist das erste Update im Binary-Log.

Der Master weiß nicht, wie viele Diener er hat, und es kümmert ihn auch nicht. Er lässt jeden Rechner teilhaben, sodass sich beliebig viele Slaves aufsetzen lassen. Ein wichtiges Detail: Ein Slave kann sogar unerwartet ausfallen. Sobald er wieder einsatzbereit ist, nimmt er Kontakt mit dem Master auf und holt selbstständig alle Updates ab. Es fällt kein Verwaltungsaufwand an.

Obwohl der Master kein eigenes Housekeeping für die Slaves betreibt, lassen sich auf ihm Informationen über die Slaves einholen. Dazu dient dem Anwender das MySQL-Kommando »show processlist« (Abbildung 1).

Das Setup beginnt

Nur mit dem aktivierten Binary-Log ist das Setup des Master-Slave-Systems noch nicht vollständig. Es beginnt schon früher. Zunächst müssen die beiden MySQL-Installationen kompatibel zueinander sein. Das ist nicht bei allen MySQL-Versionen der Fall und die Replikationen liefern bei inkompatiblen Versionen bisweilen überraschende Resultate. Ob unterschiedliche Versionen zusammenarbeiten, zeigt die Kompatibilitätstabelle auf [2].

Auf dem Master ist ein Replikations-Account anzulegen, der das Privileg »Replication Slave« bekommt. Bei MySQL-Versionen unter 4.0.2 gibt es dieses Privileg nicht. Stattdessen funktioniert dann das »File«-Privileg.

Der Account lässt sich mit »grant« folgendermaßen anlegen:

```
grant replication slave on *.* ➥
to Slave@Host.domain.de ➥
identified by 'Passwort';
```

Der Platzhalter »*Slave*« entspricht dem Namen des Accounts. »*Host.domain.de*« wird durch den FQDN (Fully Qualified Domain Name) ersetzt.

Binary-Log aktivieren

Da ein MySQL-Server standardmäßig weder Master ist noch das Binary-Log verwendet, muss man nun beides aktivieren. Dazu dienen zwei Optionen in der Datei »my.cnf«:

```
[mysqld]
log-bin
server-id=1
```

Die Option »log-bin« aktiviert das Logging. Die Server-ID ist nur eine Nummer, die den Server identifiziert. Die ID ist egal, muss aber eine positive Zahl sein. Nach diesen Änderungen ist der MySQL-Server neu zu starten.

Anschließend sind die Tabellen an der Reihe: Sie müssen geschlossen sein und alle Daten, die sich im Cache befinden, müssen ebenfalls auf die Platte geschrieben werden. Wichtig ist zudem, dass Schreibzugriffe blockiert sind. Das alles erreicht folgendes Kommando am MySQL-Prompt:

```
flush tables with read lock;
```

Sehr wichtig ist ab jetzt, dass der MySQL-Client, der gerade den Schreibzugriff blockiert, aktiv bleibt. Der Schreibschutz hebt sich sonst auf – und das darf auf keinen Fall geschehen.

Snapshot anlegen

Nun gilt es, einen Snapshot der Daten auf dem Master anzulegen. Das geht einfach mit Tar, das ein komplettes Datenverzeichnis in ein Archiv packt. Im Datenverzeichnis der MySQL-Installation gelingt dies mit dem Kommando:

```
tar -cvf /tmp/master-snapshot.tar
```

Eine Kopie der Tar-Datei kommt in das Datenverzeichnis des Slave, die sich dort mit dem Kommando

```
tar -xvf master-snapshot.tar
```

entpackt. Eine weitere Kopie sollte für Notfälle in einem sicheren Verzeichnis warten oder besser auf CD bereitliegen. Mit Hilfe dieser Tar-Datei lassen sich zu einem späteren Zeitpunkt weitere Slaves anlegen, da ja nur der Datenbestand beim Start des Binary-Log für den Slave ausschlaggebend ist.

Position des Binary-Log ermitteln

Neben dem Tar-Archiv ist die aktuelle Position des Binary-Log auf dem Master wichtig, denn nur zusammen mit dieser Information lassen sich Slaves anwerfen. Die Position fördert der Show-Befehl zutage (Abbildung 2):

```
show master status;
```

In einer kleinen Datei festgehalten sollte die Position mit auf die Backup-CD. Nun

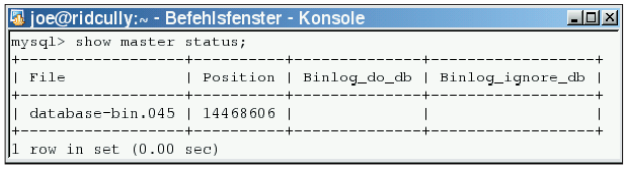

Abbildung 2: Der Show-Befehl findet die Position des Binary-Log.

sind alle wichtigen Informationen zusammen. Der Schreibschutz auf dem Master ist ebenfalls wieder gestattet und somit das Schloss vor den Tabellen nicht mehr nötig:

```
unlock tables;
```

Der Slave-Rechner ist noch mit einer eigenen Server-ID zu versorgen. Sie findet ihrem Platz in der Datei »my.cnf«. Wichtig ist, dass Master und alle Slaves unterschiedliche IDs bekommen.

```
[mysqld]
server-id=2
```

Slave zum Update auffordern

Der Slave ist jetzt ebenfalls bereit, also kann das Live-Update beginnen. Dazu sind zwar mehrere Befehle nötig, aber der Admin muss sie jeweils nur ein einziges Mal am MySQL-Prompt eingeben, und zwar:

```
change master to
 -> MASTER_HOST=Master-Name,
 -> MASTER_USER=Slave,
 -> MASTER_PASSWORD=Passwort,
 -> MASTER_LOG_FILE=Log-File,
 -> MASTER_LOG_POS=Offset;
```

Der Platzhalter »*Master-Name*« ist einfach mit dem Rechnernamen des Masters auszutauschen. Der »*Slave*« steht für dem Namen des Replikations-Accounts. Als »*Log-File*« ist der Name des Binary-Logs aus dem Snapshot anzugeben. Im Beispiel war das »database-bin.045«. Der »*Offset*« schließlich entspricht der Position des Binary-Log an der Beispielstelle »14468606«.
Jetzt muss noch der Slave-Thread auf dem Slave starten. Auch dies ist nur ein Mal erforderlich. Der MySQL-Server merkt sich, dass er als Slave fungiert. Nach dem MySQL-Kommando

```
start slave;
```

nimmt der Slave direkt Kontakt mit dem Master-Server auf und holt alle Updates ab. Bei einigen MySQL-Versionen heißt das Kommando »slave start«. Wie auch immer: Das Master-Slave-System ist jetzt fertig konfiguriert.

Datenbestand synchronisieren

Funktioniert der Master-Slave-Betrieb normal, besitzt der Slave exakt die gleichen Daten wie der Master. Nach der Installation eines neuen Slave, der erst alle Daten holen muss, ist dies nicht so – die Master-Datenbank läuft schon längst wieder und füllt sich ständig. Es hängt vom Datenbestand und dem Unterschied zwischen Snapshot und aktuellem Stand ab, wie lange die erste Synchronisation dauert. Um die Prozedur nicht zu verlängern, sollte der Master erst dann Daten annehmen, wenn der Slave seine Daten komplett übernommen hat.
Dazu bekommt der Master zuerst wieder eine Schreibsperre verpasst. Anschließend sind die aktuellen Daten des Binary-Log zu ermitteln:

```
flush tables with read lock;
show master status;
```

Nun ist der Slave mit der Aktualisierung dran, was mit folgendem Select-Statement geschieht:

```
select master_pos_wait (Arg1,Arg2);
```

Das Argument »*Arg1*« ist der Name der Logdatei, die der Befehl »master status« ausgibt; »arg2« entspricht der Position (Abbildung 2). Der Slave synchronisiert seinen Datenbestand dann mit dem Master und erst dann, wenn er fertig ist, kommt das »select«-Statement zurück. Jetzt darf auch der Master wieder Schreibzugriffe annehmen:

```
unlock tables;
```

Passend zu »show master status« gibt es auch ein »show slave status«-Kommando (Abbildung 3), das die Position, den Account und den Namen der Logdatei ausgibt.

```
joe@ridcully:~ - Befehlsfenster - Konsole
mysql> show slave status;
+--------------------+-------------+-------------+---------------+------------------+----------+
| Master_Host        | Master_User | Master_Port | Connect_retry | Log_File         | Pos      |
+--------------------+-------------+-------------+---------------+------------------+----------+
| database.nickles.de | repl       | 3306        | 60            | database-bin.045 | 14499024 |
+--------------------+-------------+-------------+---------------+------------------+----------+
1 row in set (0.00 sec)
```

Abbildung 3: Auf eine Statusabfrage per Show-Befehl liefert auch der Slave die wichtigsten Daten zurück.

Auch wenn die Replikation nicht gelingen will, hinterlassen die fehlgeschlagenen Versuche immerhin Meldungen im Error-Log. Meist handelt es sich dabei um einen Vertipper bei Account-Namen oder Passwort. Das ist mit einem neuen »change master to«-Befehl leicht zu korrigieren. Geht gar nichts mehr, bringt ein »reset slave« den Slave in den Ausgangszustand zurück.
Bei Problemen mit der Synchronisation kann es helfen, das Binary-Log zurückzusetzen. Dazu dient das Kommando »reset master«. Solche Fehler erscheinen, wenn das Binary-Log aktiv ist, obwohl der Rechner nicht als Master fungiert.

Feintuning angesagt

Das vorgestellte Replikations-Setup dient nur als Basis einer individuellen MySQL-Umgebung – zu tun gibt es noch reichlich. Per SSL (Secure Socket Layer) ist die Kommunikation zwischen Master und Slaves leicht zu verschlüsseln, was deutlich höhere Sicherheit bedeutet. Falls sich mehrere Tabellen in einer Datenbank tummeln, lassen sich bestimmte Tabellen gänzlich von der Replikation ausschließen respektive einbinden. Das funktioniert auch Datenbank-übergreifend sowie mit verschiedenen kompletten Datenbanken.
Mit zwei interessanten Änderungen beim Master- und Slave-Modus wartet die MySQL-Version 4 gegenüber den 3.23-Versionen auf: Hat sich beim Master ein ernstes Problem eingenistet, wandelt sich einer der Slaves nach einem Abstimmungsverfahren zu einem Master um. Die Lastverteilung soll zudem mit Agent-Prozessen, die selbstständig Select-Anfragen an die Slaves senden, effizienter arbeiten. *(jre)* ■

Infos

[1] Th. Wölfer, „ MySQL-Datenbanken sichern“, Teil 1: Linux-Magazin 5/04, S. 60
[2] MySQL-Kompatibilität: [http://www.mysql.com/doc/en/Replication_Compatibility.html]

Der Autor

Thomas Wölfer ist seit 20 Jahren Software-Entwickler und administriert die Mail-, Web- und Datenbankserver des Portals Nickles.de