

KAPITOLA 8

Replikace

Zabudované vybavení, které má MySQL pro potřeby replikací, tvoří základy pro budování rozsáhlých, vysoce výkonných aplikací nad MySQL. Replikace umožňuje nakonfigurovat jeden nebo více serverů jako repliky ("otroky", slaves) jiného serveru. To není prospěšné pouze pro vysoce výkonné aplikace – hodí se také pro mnohé jiné úlohy, například pro sdílení dat se vzdáleným pracovištěm, pro "zahrátou" nakonfigurovanou rezervní jednotku, které se říká hot spare, nebo když udržujete server s kopií ostrých dat pro testovací či výukové účely. V této kapitole prozkoumáme všechny aspekty replikace. Začneme přehledem, jak funguje, pak se podíváme na základní přípravu serveru, na navrhování pokročilejších replikačních konfigurací, a neopomeneme správu a optimalizaci replikovaných serverů. Přestože se v této knize hlavně soustředíme na výkon, v případě replikace se budeme stejnou měrou zabývat i bezchybností a spolehlivostí, takže si ukážeme, jak připravit replikaci tak, aby fungovala dobře. Podíváme se také na některé nadcházející změny a zdokonalení v replikaci MySQL, mezi něž patří několik zajímavých záplat, které vytvořil v Google.

Replikace přehledně

Základní problém, který řeší replikace, je udržet data jednoho serveru synchronizovaná s daty jiného serveru. K jedinému "pánovi", master serveru, nebo zkráceně masteru, se může připojit několik "otroků", slave serverů, zkráceně replik, a replika zase může naopak pracovat jako master. Masters a repliky můžete uspořádat v mnoha různých topologiích. Můžete replikovat celý server, replikovat jen některé databáze, nebo dokonce určovat, které tabulky se mají replikovat.

MySQL podporuje dva druhy replikace: příkazovou a řádkovou. Příkazová (neboli "logická") replikace je k dispozici už od MySQL 3.23 a v ostrém provozu ji používá většina lidí. Řádková replikace je novinkou v MySQL 5.1. Oba druhy replikace pracují tak, že zaznamenávají změny v binárním logu mastera¹ a přehrávají tento log na repliku. Obě jsou asynchronní – tj. u kopie dat na replice

¹ Je-li pro vás pojem binárního logu nový, naleznete více informací v kapitole 6, v této kapitole, a v kapitole 11.

není zaručeno, že se v každém okamžiku bude jednat o "nejčerstvější data"¹. Není zde ani žádná garance toho, jak dlouhá může být latence na replice. Rozsáhlé dotazy mohou způsobovat, že budou repliky zpožděny v řádu sekund, minut, nebo dokonce i hodin za masterem.

Replikace v MySQL je většinou zpětně kompatibilní. Tj. novější server obvykle může bez jakýchkoliv potíží sloužit jako replika staršího serveru. Ovšem starší verze serveru často nemohou sloužit jako repliky novějších verzí – nemusejí rozumět některým novým funkcím, nebo nové syntaxi SQL, jež používá novější server, a mohou zde být i odlišnosti ve formátech souborů, které se při replikaci používají. Například, nemůžete replikovat z mastera MySQL 5.0 na repliku MySQL 4.0. Vždy se vyplatí otestovat replikační konfiguraci, než podniknete upgrade z jedné hlavní verze na jinou, jako z 4.1 na 5.0, nebo z 5.0 na 5.1.

Replikace pro mastera obecně neznamená mnoho režie navíc. Požaduje sice, aby na masteru bylo zapnuto zaznamenávání do binárního logu, které může mít významnou režii, ale pro řádné zálohování je stejně potřebujete. Kromě binárního logu přidává během normálního fungování něco režie mastera i každá připojená replika (většinou se jedná o síťový I/O).

Replikace je poměrně dobrá pro škálování čtení, protože je můžete směřovat na repliku, ale není už tak dobrá pro škálování zápisů, pokud ji nenavrhnete opravdu dobře. Když k masteru připojíte hodně replik, jednoduše to způsobí, že zápisy se budou muset udělat mnohokrát, na každé replice jednou. Celý systém pak bude omezen počtem zápisů, které umí vykonat nejslabší část systému.

Replikace také znamená plýtvání, pokud používáte více než několik replik, protože se v podstatě zbytečně duplikuje spousta dat. Například, jediný master s 10 replikami znamená 11 kopií stejných dat a duplikaci většiny stejných dat v 11 různých cache. Je to analogické s 11 disky spojenými do RAID 1 na úrovni serveru. Ačkoliv to není hospodárné využití hardwaru, přesto až překvapivě často vidíme tento druh replikační konfigurace. Různé způsoby, jak se dá tento problém zmírnit, probíráme na různých místech této kapitoly.

Problémy, které řeší replikace

Replikace se běžně používá při řešení následujících problémů:

- **Distribuce dat.** Replikace MySQL obvykle nezatěžuje příliš šířku pásma² a můžete ji podle chuti zastavovat a startovat. Je užitečná pro udržování kopie dat v zeměpisně vzdálených lokalitách, jako jsou různá datová centra. Vzdálená replika může dokonce pracovat s připojením, které je přerušované (neúmyslně nebo jinak). Pokud ovšem chcete, aby vaše repliky měly velmi malé replikační zpoždění, budete potřebovat stabilní spoj s nízkou latencí.
- **Rozložení zátěže (load balancing).** Replikace MySQL může pomoci s distribucí čtecích dotazů přes několik serverů, funguje velmi dobře u aplikací, které intenzivně čtou. Základního

1 Více k tomu viz "Synchronní MySQL replikace" na straně 476.

2 Jak uvidíte později, řádková replikace, která byla zavedena v MySQL 5.1, může používat mnohem více šířky pásma než tradičtější příkazová replikace.

rozložení zátěže lze docílit s několika jednoduchými změnami kódu. V malém měřítku se dají použít zjednodušené přístupy, jako explicitně kódované názvy hostitelů nebo DNS používané "pořád dokola" (ve stylu "round-robin", které pomocí jediného názvu hostitele ukazují na několik IP adres). Nebo použijte sofistikovanější přístupy. Chcete-li zátěž distribuovat mezi MySQL servery, budou dobře fungovat standardní řešení pro rozložení zátěže, jako jsou různé produkty pro síťové rozložení zátěže. Dobře také funguje projekt Linux Virtual Server (LVS). Rozložení zátěže budeme detailněji probírat v kapitole 9.

- **Zálohování.** Replikace je cenná technika, která může vypomoci se zálohováním, nicméně rozhodně nechápejte repliku ani jako zálohu, ani jako náhradu záloh.
- **Vysoká dostupnost a rychlá náhrada vypadlého serveru (failover).** Replikace může zajistit, aby se MySQL stal tzv. kritickým místem výpadku (SPOF, single point of failure), neboli aby kvůli němu vypadla celá aplikace. Dobrý systém náhrady serverů, které vypadnou (failover system), jenž zahrnuje replikované servery, může pomoci významně zredukovat prostoje. Toto téma rovněž probereme podrobněji v kapitole 9.
- **Testování při upgradu MySQL.** Je běžnou praxí připravit repliku s upgradovanou verzí MySQL a nějakou dobu ji používat, abyste se přesvědčili, že dotazy pořád pracují tak, jak očekáváte. Teprve poté upgradujete všechny ostatní instance.

Jak funguje replikace

Než se pustíme do detailů přípravy replikace, podívejme se, jak MySQL skutečně replikuje data. Na vysoké úrovni je replikace jednoduchý postup složený ze tří kroků:

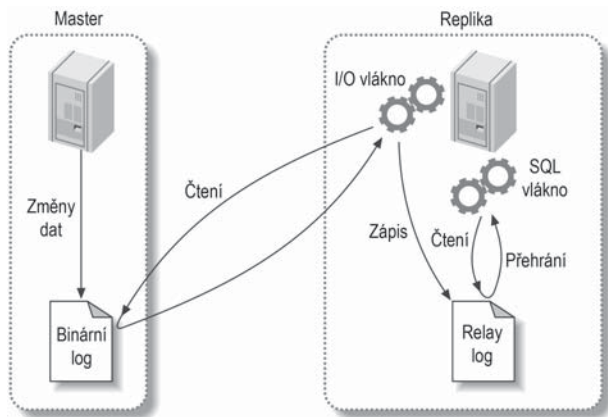
1. Master zaznamená změny ve svých datech do svého binárního logu. (Těmto záznamům se říká události binárního logu, binary log events.)
2. Replika zkopíruje události binárního logu masteru do svého logu, říká se mu relay log.
3. Replika přehraje události nacházející se v relay logu a aplikuje změny na svá vlastní data.

Uvědomte si, že tohle je pouze přehled – každý z těchto kroků je dost složitý. Podrobněji je replikace znázorněna na obrázku 8-1.

První částí postupu je zaznamenávání do binárního logu na masteru (později vám ukážeme, jak tohle připravit). Těsně předtím, než se na masteru dokončí jakákoliv transakce, která aktualizuje data, master zaznamená její změny do svého binárního logu. MySQL zapisuje jednotlivé transakce do binárního logu jednu po druhé, i když se příkazy v transakcích během vykonávání střídaly. Poté, co master zapsal události do binárního logu, sdělí úložným enginům, aby transakce potvrdily.

V dalším kroku zkopíruje replika binární log mastera na svůj vlastní pevný disk, do relay logu. Začne tím, že nastartuje zpracovatelské vlákno, kterému se říká I/O vlákno repliky (I/O slave thread). Toto I/O vlákno otevře obyčejné klientské připojení k masteru a následně nastartuje speciální proces binlog dump (není k němu odpovídající příkaz SQL). Proces binlog dump čte události z binár-

ního logu masteru. Nedotazuje se na události. Jakmile dostihne mastera, jde spát a čeká na signál od mastera, že jsou k dispozici nové události. I/O vlákno zapíše události do relay logu repliky.



Obrázek 8-1. Jak funguje replikace MySQL.

Před MySQL 4.0 fungovala replikace v mnoha ohledech úplně jinak. Například – první replikační funkcionalita MySQL nepoužívala relay log, což znamenalo, že replikace používala jen dvě vlákna, ne tři. Protože většina lidí provozuje novější verze serveru, v této kapitole už nebudeme zmiňovat jakékoliv další detaily o velmi starých verzích MySQL.

Poslední část celého postupu zpracovává SQL vlákno repliky (SQL slave thread). Toto vlákno čte a přehrává události z relay logu, což znamená, že aktualizuje data repliky tak, aby se shodovala s daty masteru. Dokud toto vlákno udrží krok s I/O vláknem, relay log obvykle setrvává v cache operačního systému, takže relay logy mají velmi nízkou režii. Události, které vykonává SQL vlákno repliky, mohou volitelně jít do vlastního binárního logu repliky, což se hodí ve scénářích, jež budeme zmiňovat později v této kapitole.

Na obrázku 8-1 jsou znázorněna pouze dvě replikační vlákna, která běží na replice, nicméně ještě existuje jedno vlákno na masteru: podobně jako jakékoliv jiné připojení k MySQL serveru, to připojení, jež replika otevře k masteru, nastartuje vlákno na masteru.

Tato replikační architektura odděluje na replice procesy získávání a přehrávání událostí, takže mohou být asynchronní. Jinak řečeno: I/O vlákno může pracovat nezávisle na SQL vláknem. Vnáší také do replikačního postupu jistá omezení – nejdůležitější z nich je to, že na replice je replikace serializována. To znamená, že aktualizace, které možná na masteru probíhaly paralelně, v různých vláknech, nemohou být na replice paralelizovány. A jak uvidíme později, právě tohle tvoří úzké hrdlo výkonu pro mnohé pracovní zátěže.

Příprava replikace

Ačkoliv příprava replikace je v MySQL poměrně jednoduchá, existuje mnoho různých variant základních kroků, záleží na konkrétním scénáři. Základním scénářem je čerstvě nainstalovaný master a replika. Na vysoké úrovni vypadá postup takto:

1. Na každém ze serverů se připraví replikační účet.
2. Nakonfiguruje se master a replika.
3. Replíce se vydá pokyn, aby se připojila k masteru a odtud replikovala.

Zde se předpokládá, že mnoho výchozích hodnot bude postačujících, což je pravda, pokud jste master a repliku právě nainstalovali a mají stejná data (výchozí databázi `mysql`). Postupně si předvedeme všechny kroky a budeme předpokládat, že se servery jmenují `server1` (IP adresa 192.168.0.1) a `server2` (IP adresa 192.168.0.2). Poté si vysvětlíme, jak se inicializuje replika ze serveru, který je už v provozu, a prozkoumáme doporučenou replikační konfiguraci.

Vytvoření replikačních účtů

MySQL má několik speciálních oprávnění, která umožňují běh replikačních procesů. I/O vlákno repliky, které běží na replikačním serveru repliky, učiní TCP/IP připojení k masteru. To znamená, že na masteru musíte vytvořit nový uživatelský účet a přidělit mu patřičná oprávnění, aby se I/O vlákno mohlo připojit jako tento uživatel a číst binární log mastera. Podívejte se, jak se vytvoří takový uživatelský účet, který jsme pro účely této ukázky nazvali `repl`:

```
mysql> GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.*  
-> TO repl@'192.168.0.%' IDENTIFIED BY 'p4ssword';
```

Tento účet vytvoříme na masteru i na replíce. Připomínáme, že jsme uživatele omezili na lokální síť, protože replikační účet není bezpečný. (Další informace k bezpečnosti viz kapitola 12.)

Replikační uživatel ve skutečnosti potřebuje na masteru jen oprávnění REPLICATION CLIENT, nepotřebuje oprávnění REPLICATION SLAVE ani na jednom ze serverů. Tak proč jsme toto oprávnění udělili na obou serverech? Ze dvou následujících důvodů:

- Účet, se kterým budete monitorovat a spravovat replikaci, potřebuje oprávnění REPLICATION SLAVE a je mnohem praktičtější, když se použije jediný účet pro tyto účely, než pro ně vytvářet samostatný uživatelský účet.
- Připravíte-li účet na masteru a poté z něho naklonujete repliku, bude replika správně připravena, aby mohla fungovat jako master, pro případ, že byste chtěli prohodit role repliky a mastera.

Konfigurace mastera a repliky

Dalším krokem je zapnout několik nastavení na masteru, o němž předpokládáme, že se jmenuje `server1`. Je potřeba, abyste zapnuli zaznamenávání do binárního logu a specifikovali ID serveru. Zadejte do souboru `my.cnf` mastera následující řádky (nebo ověřte, zdali tam jsou):

```
log_bin = mysql-bin
server_id = 10
```

Přesné hodnoty záleží na vás. Ačkoliv my se zde vydáváme tou nejschůdnější cestou, vy si můžete pochopitelně připravit něco propracovanějšího.

Musíte explicitně specifikovat jedinečné ID serveru. Zvolili jsme 10, nikoliv 1, protože 1 je výchozí hodnota, kterou server typicky vybere, když není specifikována žádná hodnota. (Je to závislé na verzi; některé verze MySQL jednoduše vůbec nebudou pracovat.) Hodnota 1 tak může snadno zmást a být v konfliktu se servery, které nemají svá explicitní ID. Běžnou praxí je použít poslední oktet IP adresy serveru, za předpokladu, že se nebude měnit a že je jedinečný (tj. server spadá pouze do jedné podsítě).

Pokud ještě není v konfiguračním souboru masteru specifikováno zaznamenávání do binárního logu, budete muset restartovat MySQL. Abyste prověřili, zdali je na masteru vytvořený soubor binárního logu, vydejte příkaz `SHOW MASTER STATUS` a zkontrolujte, zdali jste dostali výstup obdobný následujícímu (MySQL přidá na konec názvu souboru nějaké číslice, takže neuvídíte soubor s přesně stejným názvem, který jste specifikovali):

```
mysql> SHOW MASTER STATUS;
```

```
+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-bin.000001 |      98 |              |                  |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Replika potřebuje ve svém konfiguračním souboru `my.cnf` něco podobného, jako má master. Nezapomeňte, že i na replice bude potřeba restartovat MySQL:

```
log_bin           = mysql-bin
server_id         = 2
relay_log         = mysql-relay-bin
log_slave_updates = 1
read_only         = 1
```

Některé z těchto voleb nejsou v zásadě nutné a u jiných jsme prostě jen explicitně uvedli výchozí hodnotu. Ačkoliv se v praxi na replice vyžaduje pouze parametr `server_id`, zapnuli jsme také `log_bin` a specifikovali explicitní název pro soubor binárního logu. Výchozí název se totiž bere z hostitelského názvu serveru, což může dělat potíže, jestliže se hostitelský název změní. Dále