Duplikacja i replikacja MySQL

Opracowanie:
Piotr Knopczyński
Marcin Talarczyk

Replikacja

Replikacja danych to proces powielania informacji między różnymi serwerami baz danych.

Replikacja pozwala nam na:

- **Skalowalność –** możliwe jest rozłożenie obciążenia pomiędzy wieloma serwerami. Operacje zapisu i aktualizacji rekordów odbywają się na jednym serwerze, a pobieranie i przeszukiwanie danych z drugiego.
- **Bezpieczeństwo –** dzięki replikacji tworzymy klona istniejącej bazy produkcyjnej. Nie uchroni nas to przed operacjami typu `DROP TABLE`, ale może pomóc w przypadku awarii sprzętowej głównego serwera.
- **Analiza** – skomplikowane operacje analityczne, różnego rodzaju przeliczenia i analizy statystyczne mogą być wykonywane na osobnym serwerze bez konieczności obciążania głównej bazy.
- **Separacja** – możemy udostępnić klon bazy produkcyjnej dla developerów lub testerów, aby wykonali swoje prace na kopii bazy.

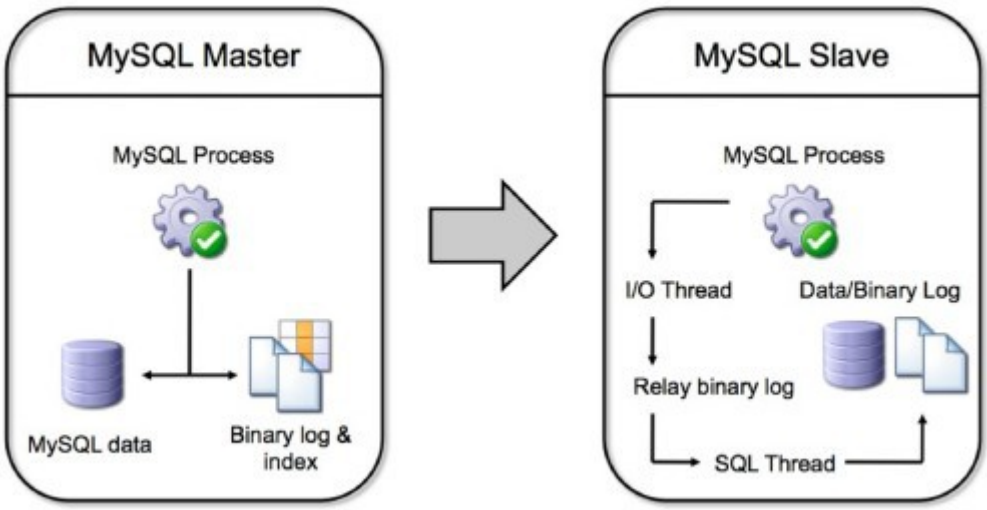
Mechanizmy replikacji MySQL

Replikacja danych MySQL opiera się o bardzo prostą zasadę. Serwer główny (master), prowadzi swego rodzaju dziennik, w którym zapisuje każdą czynność jaką wykonał. Wykorzystuje do tego bin-logi. Są to pliki binarne zawierające instrukcje jakie wykonał master. Serwer zapasowy (slave) odczytuje te dane i kolejno wykonuje zapytania zapełniając bazę kolejnymi rekordami. Efektem tej pracy są dwie identyczne bazy danych.

Po skonfigurowaniu mechanizmu replikacji na serwerze master pojawia się dodatkowy wątek, który odpowiada za wysyłanie bin-logów do serwerów slave. Z kolei serwer zapasowy tworzy dwa wątki:

- **I/O Thread –** odpowiada za odbieranie dziennika od serwera głównego i zapisuje go w plikach tymczasowych relay-log.
- **SQL Thread** – zajmuje się parsowaniem tych plików i wykonywaniem zapytań do bazy.

Całą sytuację przedstawia poniższy schemat:



Rodzaje replikacji

System bazodanowy MySQL umożliwia trzy różne metody replikacji, co przekłada się na format danych zapisywanych do bin-logów. Za wybór metody relikacji odpowiada zmienna `binlog_format`, która może przyjąć wartość: `ROW`, `STATEMENT`, `MIXED`.

Metody replikacji:

- **SBR** (statement-based replication) - w tym trybie, serwer do pliku zapisuje zapytania jakie wykonał.
- **RBR** (row-based replication) - do bin-logów zapisywane są wyniki działań zapytań na serwerze master. Zapisywana jest informacja jaki rekord został w jaki sposób zmieniony.
- **MFL** (mixed-format logging) - jest to połączenie dwóch powyższych typów replikacji.

Technika replikacji **SBR** jest bardzo szybka i wydajna, ponieważ w jej przypadku serwer główny zapisuje do pliku zapytanie jakie wykonał, następnie serwer zapasowy je odczytuje i wykonuje. Niestety do pliku logów zapisywane są tylko zapytania SQL, co przysporzy nam problemów, gdy nasze zapytania będą bardziej złożone.

Problem ten rozwiązała metoda **RBR**, która do bin-logów zapisuje wyłącznie zmiany jakie zaszły po wykonaniu polecenia - logowane są informacje na temat sposobu modyfikacji konkretnych rekordów. Niestety metoda ta jest znacznie wolniejsza od poprzedniej oraz zwiększa ilość wysyłanych danych pomiędzy replikującymi się serwerami.

Z pomocą przyszła nam metoda **MFL**, w której w większości przypadków, logowane są zapytania SQL tak jak w przypadku **SBR**, natomiast dla zapytań, których wynik nie jest przewidywalny, włączana jest replikacja **RBR**.

Konfiguracja

Do konfiguracji użyjemy dwóch maszyn ze świeżo zainstalowanym pakietem *mysql* i *mysql-server*. Załóżmy, że nasze serwery mają następujące IP:

- **master** => 1.1.1.1
- **slave** => 1.1.1.2

W pliku **/etc/mysql/my.cnf** w sekcji [mysqld] dodajemy następujące linie:

```
server-id = 1
log-bin = /var/log/mysql/mysql-bin.log
sync_binlog = 1
```

oraz na maszynie **slave**:

```
server-id = 2
log-bin = /var/log/mysql/mysql-bin.log
sync_binlog = 1
```

Zmienna **server-id** musi mieć unikalną wartość na każdym z serwerów oraz być różna od 0. Chcemy też by MySQL automatycznie usuwał pliki mysql-bin.log starsze niż 10 dni, a maksymalna wielkość pojedynczego pliku to 500MB. Warto też się upewnić czy wartość opcji **innodb_flush_log_at_trx_commit** jest równa 1.

```
expire_logs_days     = 10
max_binlog_size      = 500M
innodb_flush_log_at_trx_commit = 1
```

Zmieniamy również wartość zmiennej **bind-addres**, żeby serwer mysql nasłuchiwał na wszystkich adresach IP:

```
bind-address            = 0.0.0.0
```

Możemy teraz zrestartować proces **mysqld**:

```
[root@master ~]# /etc/init.d/mysqld restart
[root@slave ~]# /etc/init.d/mysqld restart
```

Na serwerze **master** dodamy nowego użytkownika z prawami do replikacji:

```
mysql> CREATE USER 'slaveuser'@'%' IDENTIFIED BY 'haslo';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'slaveuser'@'%';
mysql> flush privileges;
```

Możemy przetestować połączenie z serwera **slave**:

```
[root@slave ~]# mysql -u slaveuser -h 1.1.1.1 -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 979
Server version: 5.5.25-log MySQL Community Server
...
```

Pobranie współrzędnych pliku replikacji

Wraz z włączeniem replikacji *mysql* tworzy pliki dziennika w katalogu **/var/log/mysql** (dokładnie tak jak zdefiniowaliśmy w pliku **/etc/mysql/my.cnf**). Do synchronizacji serwera slave z master, musimy znać aktualną pozycję tego pliku. Musimy też zablokować możliwość dodawania/edycji/usuwania danych z naszej bazy na pewien czas, ponieważ każda z tych operacji powoduje zmianę współrzędnych. Na serwerze **master** wpisujemy:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> show master status\G;
*************************** 1. row ***************************
File: mysql-bin.000001
Position: 107
Binlog_Do_DB:
Binlog_Ignore_DB:
```

Jak widzimy powyżej, aktualny plik dziennika to **mysql-bin.000001**, a pozycja ma wartość = 107.

Eksport i import baz danych

Kolejnym krokiem jest eksport bazy z serwera **master** na **slave**.

```
[root@master ~]# mysqldump --add-drop-table --master-data --quick -u
superroot -p --all-databases > /tmp/db-master.sql
```

Po zrzuceniu bazy, możemy odblokować tabele:

```
mysql> UNLOCK TABLES;
```

Następnie kopiujemy plik **db-master.sql** na serwer **slave**:

```
[root@master ~]# scp /tmp/db-maser.sql root@1.1.1.2:/tmp
```

oraz importujemy ją (przedtem musimy zatrzymać proces **slave**):

```
mysql> STOP SLAVE;
mysql> source /tmp/db-master.sql;
```

Konfiguracja serwera slave

Ostatnim krokiem jest „powiedzenie” serwerowi **slave**, gdzie jest znajduje się **master**:

```
mysql> CHANGE MASTER TO
MASTER_HOST='1.1.1.1',
MASTER_USER='slaveuser',
MASTER_PASSWORD='haslo',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=107;
```

Gdzie **MASTER_LOG_FILE** i **MASTER_LOG_POS** to współrzędne dziennika replikacji, które pobraliśmy wcześniej (show master status). Teraz możemy ponownie uruchomić proces **slave** oraz sprawdzić jego status:

```
mysql> start slave;
mysql> show slave status\G;
*************************** 1. row ***************************
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

Jeżeli wartość **Slave_IO_Running** oraz **Slave_SQL_Running** jest ustawiona na ***Yes*** oznacza to, że wszystko działa. Możemy teraz przetestować replikację, zmieniając coś w bazie na serwerze **master**. Zmiana powinna być natychmiast widoczna na serwerze **slave**. Należy też pamiętać, że zmiany wykonane tylko na serwerze **slave** mogą prowadzić do błędów oraz zatrzymania replikacji.

Plusy i minusy replikacji

Plusy:

- skalowalność
- bezpieczeństwo – tworzenie klona bazy produkcyjnej
- analiza
- separacja

Minusy:

- nie uchroni nas przed operacjami typu DROP TABLE
- nie daje nam żadnej gwarancji, że po wykonaniu operacji dane z serwera głównego zostaną prawidłowo przesłane na serwer zapasowy

Duplikacja

Kiedy wiemy już na czym polega i jak skonfigurować replikacje baz danych w mysql, czas powiedzieć parę słów na temat duplikacji. W najprostszych słowach jest to replikacja „master-master” (w odróżnieniu od replikacji master-slave opisanej wcześniej). Oznacza to, że zamiast rekonstruowania zmian wprowadzonych na po stronie mastera na serwerze slave, dochodzi tutaj do obustronnej synchronizacji między bazami danych. Dzięki takiemu rozwiązaniu zmiany wprowadzone w jednej z dwóch baz danych zostaną wprowadzone również w drugiej (przeprowadzanie transakcji jest możliwe na obu serwerach, a nie tylko na jednej jak w przypadku master-slave).

Konfiguracja

Wiemy już co chcemy uzyskać. Pora odpowiedzieć na pytanie w jaki sposób zmusić dwie bazy danych do takiej współpracy? Część czynności wykonuje się w sposób identyczny jak w przypadku konfiguracji master-slave, a resztę w sposób analogiczny.

Znowu do przedstawienia konfiguracji użyjemy dwóch maszyn wirtualnych zainstalowanymi pakietami mysql i założymy, że nasze serwery mają następujące IP:

- **master1, slave2** => 1.1.1.1
- **master2, slave1** => 2.2.2.2

Ponownie zaczynamy od edycji pliku **/etc/mysql/my.cnf.** Odpowiednio dla:

master1, slave2:

```
server-id = 1
log-bin = /var/log/mysql/mysql-bin.log
sync_binlog = 1
master-host = 2.2.2.2
master-user = slaveuser
master-password = haslo
```

master2, slave1:

```
server-id = 2
log-bin = /var/log/mysql/mysql-bin.log
sync_binlog = 1
master-host = 1.1.1.1
master-user = slaveuser
master-password = haslo
```

Gdzie parametry z przedrostkiem „master" odpowiadają danym potrzebnym do połączenia z serwerem master i tamtejszą bazą danych, t.j. master-host to adres IP serwera, który jest masterem tego konfigurowanego. Parametry master-user i master-password to odpowiednio nazwa i hasło użytkownika stworzonego w bazie danych do obsługi replikacji (więcej o tym w dalszej części referatu). Ponadto w pliku my.cnf możemy ustawić m.in.: automatyczne usuwanie plików mysql-bin starszych od danej ilości dni, maksymalną pojemność pliku z bin-logami, które bazy danych/tabele mają być ignorowane podczas replikacji, czy też ilość sekund jakie slave odczekuje zanim ponownie odpyta serwer master. Wszystko to dzięki następującym parametrom:

```
expire_logs_days
max_binlog_size
replicate-ignore-db
replicate-ignore-table
master-connect-retry
```

Kolejnym krokiem naszej konfiguracji będzie utworzenie użytkowników z prawami do replikacji mysql dla obu serwerów.

master1, slave2:

```
mysql> CREATE USER 'slaveuser'@'2.2.2.2' IDENTIFIED BY 'haslo';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'slaveuser'@'2.2.2.2';
mysql> flush privileges;
```

master2, slave1:

```
mysql> CREATE USER 'slaveuser'@'1.1.1.1' IDENTIFIED BY 'haslo';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'slaveuser'@'1.1.1.1';
mysql> flush privileges;
```

Teraz możemy przetestować połączenie między serwerami (przykładowo z master1 na master2):

```
root@asl1:~>mysql -u slaveuser -h 2.2.2.2 -p
Enter password:
```

Teraz został nam do zrobienia już tylko restart demona mysqld na obu serwerach:

```
root@asl1:~>services mysql restart
root@asl2:~>services mysql restart
```

Oraz sprawdzenie czy wszystko działa poprawnie. Aby to zrobić włączamy klienta mysql na obu maszynach i wpisujemy polecenie:

```
mysql> show slave status\G;
mysql> show master status\G;
```

W wyniku tego polecenia wyświetlone zostaną informacje związane z aktualnym stanem serwera w postaci następujących parametrów:

- **Slave-IO-State** - informacja w jakim stanie aktualnie znajduje się serwer slave. Np. ‘waiting for master update’, ‘connecting to master’, ‘reconnecting after a failed binlog dump request’, itd.
- **Master_Host** - adres IP serwera master
- **Master_User** - nazwa użytkownika z uprawnieniami replikacji w bazie danych serwera master
- **Master_Port** - port, na ktorym nałuchuje Master
- **Connect_ Retry** - ilość sekund, które czeka Slave zanim ponownie odpyta serwer Master.
- **Master_Log_File** - nazwa pliku binarnego z logami sql na serwerze Master.
- **Read_Master_Log_Pos** - numer pozycji pliku z logami serwera Master.
- **Relay_Log_File i Relay_Log_Pos** - to informacje, od którego miejsca Slave ma zacząć odpytywanie plików z logami serwera Master.
- **Slave_IO_Running oraz Slave_SQL_Running** - przyjmują wartości Yes lub No w zależności czy replikacja działa poprawnie (działa połączenie Master-Slave).

Ponadto **show slave status\G** wyświetla obszerną listę potencjalnych błędów jakie mogą nastąpić podczas współpracy naszych serwerów wraz z liczbą ich wystąpienia.

Możliwe problemy związane z konfiguracją duplikacji

1. w plikach konfiguracyjnych mysql aktywna jest opcja **bind-adress = 127.0.0.1** lub **skip-networking.** Należy upewnić się, że obie opcje są nieaktywne.
2. jeśli nie utworzył się katalog ***/var/log/mysql*** można zrobić to ręcznie korzystając z następujących poleceń:

```
root@asl1:/etc>mkdir /var/log/mysql
root@asl1:/etc>chown _mysql._mysql /var/log/mysql/
```

Plusy i minusy duplikacji

Plusy:

- z połączeniem z innym oprogramowaniem można zyskać na wydajności w usługach wymagających częstych zapisów do bazy danych (rozdzielamy pracę na więcej serwerów)
- dodatkowa kopia
- separacja
- skalowalność

Minusy:

- często może dochodzić do konfliktów między bazami na różnych serwerach
- podobnie jak przy zwykłej replikacji