Mysql可扩展性—(复制)Replication

MySQL的Replication是一种多个MySQL的数据库做主从同步的方案，特点是异步，广泛用在各种对MySQL有更高性能，更高可靠性要求的场合。与之对应的另一个技术是同步的MySQL Cluster，但因为比较复杂，使用者较少。

一、概述

1、概念：

MySql Replication从mysql 3开始被支持，是一种对多个mysql的数据库做主从同步的方案。通过配置各个mysql server的my.cnf实现数据的同步。

2、特点：

数据复制技术有以下一些特点：

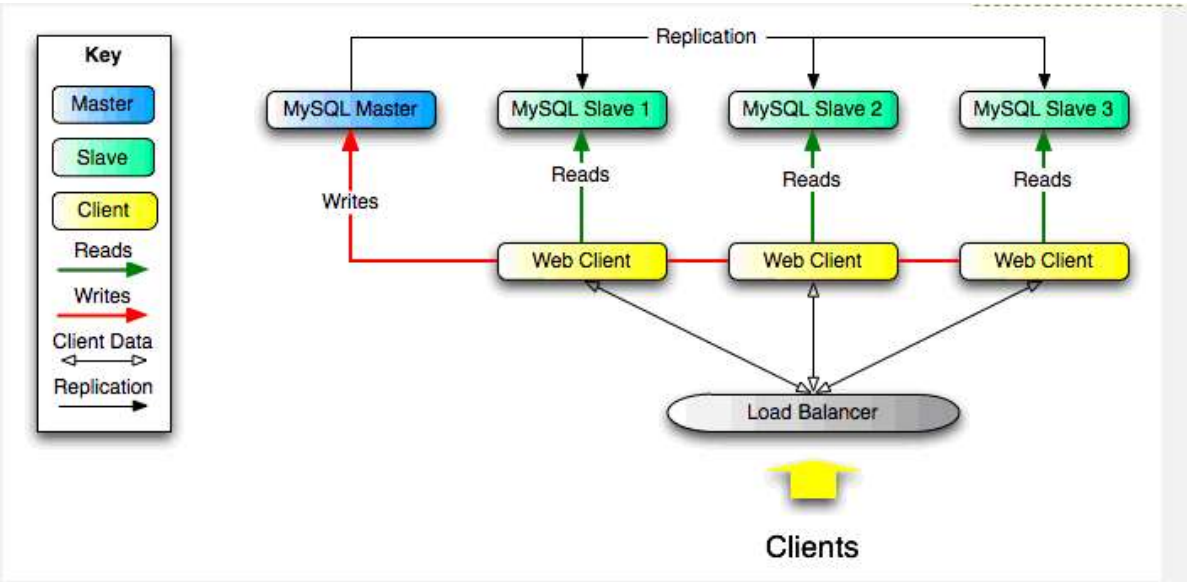
(1) 数据分布

(2) 负载平衡(load balancing)

(3) 备份

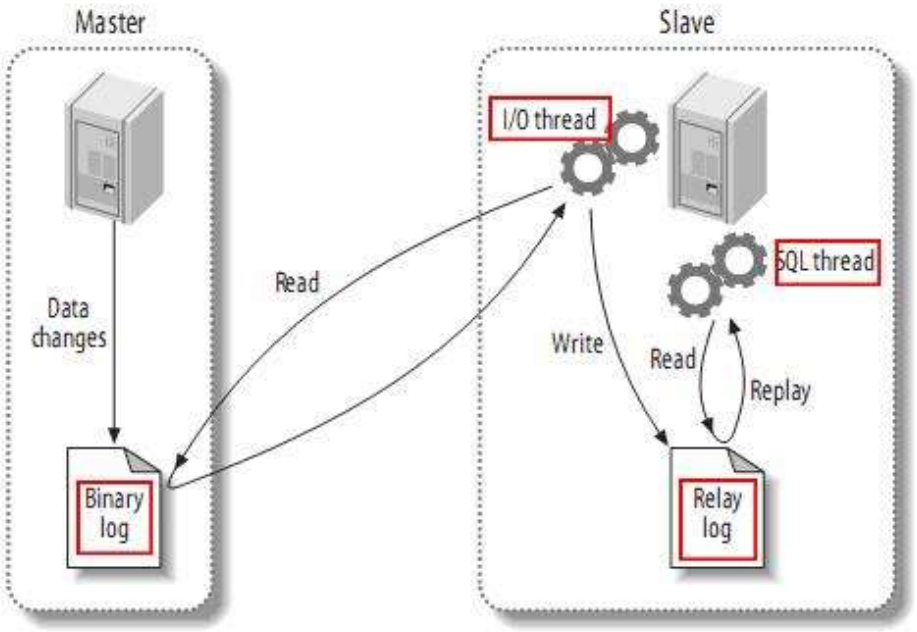
(4) 高可用性(high availability)和容错

使用场景：



3、原理

Mysql 的 Replication 是一个异步的复制过程，从一个MySQL节点（称之为 Master）复制到另一个MySQL节点（称之Slave）。在 Master 与 Slave 之间的实现整个复制过程主要由三个线程来完成，其中两个线程（SQL 线程和 I/O 线程）在 Slave 端，另外一个线程（I/O 线程）在 Master 端。



三步走：

(1) master将改变记录到二进制日志(binary log)中（这些记录叫做二进制日志事件，binary log events）；

(2) slave将master的binary log events拷贝到它的**中继日志**(relay log)；

(3) slave重做中继日志中的事件，将改变反映它自己的数据。

总结：

* 每个从仅可以设置一个主。
* 主在执行sql之后，记录二进制log文件（bin-log）。
* 从连接主，并从主获取binlog，存于本地relay-log，并从上次记住的位置起执行sql，一旦遇到错误则停止同步。

推论：

* 主从间的数据库不是实时同步，就算网络连接正常，也存在瞬间，主从数据不一致。
* 如果主从的网络断开，从会在网络正常后，批量同步。
* 如果对从进行修改数据，那么很可能从在执行主的bin-log时出现错误而停止同步，这个是很危险的操作。所以一般情况下，非常小心的修改从上的数据。
* 一个衍生的配置是双主，互为主从配置，只要双方的修改不冲突，可以工作良好。
* 如果需要多主的话，可以用环形配置，这样任意一个节点的修改都可以同步到所有节点。

二、搭建

简单的主从配置。

1、创建replication账号，通常为支持主从动态同步，或手动切换，都是在主从节点上创建好统一账号。

2、配置master和slave的my.cnf；

3、slave连接master开始复制；

3.1 创建账号repl并授予复制权限：

```
mysql> CREATE USER 'username'@'host' IDENTIFIED BY 'password';
mysql> GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO
'username'@'192.168.1.10' IDENTIFIED BY 'password';

# 或者直接使用 grant命令存在则授权，不存在则自动创建账号并授权；

# 或者 insert into mysql.user(Host,User,Password)
values("localhost","test",password("1234"));
```

用户名和密码都会存储在文本文件master.info中。

3.2 配置master，主要是打开二进制日志，指定唯一的server id（可取当前主机的ip后3位）

```
[mysqld]
# binary logging is required for replication
log-bin=mysql-bin
binlog_format=mixed         # mixed、statement、row

# required unique id between 1 and 2^32 - 1
# defaults to 1 if master-host is not set
# but will not function as a master if omitted
server-id       =  112

log_slave_updates = true    # 支持连级复制,如果master不作slave可不设置
```

重启master，service mysql restart；查看 主 状态

```
mysql> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000006 |     2392 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

配置slave，

```
[mysqld]
# binary logging is required for replication
log-bin=mysql-bin                                   # 非必须，单若slave是其他slave的
master则必须设置bin-log

# required unique id between 1 and 2^32 - 1
# defaults to 1 if master-host is not set
server-id       =  188                              # 不同于 master的 server-id
relay_log       =  mysql-relay-bin                  # 中继日志
log_slave_updates = 1                               # log_slave_updates表示slave将
复制事件写进自己的二进制日志
read_only = 1                                       # 开启了slave的二进制日志，却没有
设置log_slave_updates，然后查看slave的数据是否

                                                    # 改变，这是一种错误的配置。所以，
尽量使用read_only，它防止改变数据(除了特殊的线

                                                    # 程)。但是，read_only并是很实
用，特别是那些需要在slave上创建表的应用。

replicate_wild_do_table=repldb.%

#sync database
#replicate-do-db=platform          # 注释该行
```

重启 slave；

3.3 启动slave，让slave连接master，并开始重做master二进制中的时间。

注意：不建议在my.cnf 配置文件中直接操作，而是使用CHANGE MASTER TO 语句；该语句可以完全取代对配置文件的修改，还能为slave指定不同的master，而不需要停止服务器。

```
mysql> CHANGE MASTER TO MASTER_HOST='server1',
    -> MASTER_USER='repl',
    -> MASTER_PORT=3306,
    -> MASTER_PASSWORD='password',
    -> MASTER_LOG_FILE='mysql-bin.000001',
    -> MASTER_LOG_POS=0;

# MASTER_LOG_POS的值为0，因为它是日志的开始位置。然后，你可以用SHOW SLAVE STATUS语句查看slave的设置是否正确：
```

查看slave的设置是否正确：

```
mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 120.24.49.112
                  Master_User: repl
                  Master_Port: 16810
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000006
          Read_Master_Log_Pos: 875
               Relay_Log_File: mysql-relay-bin.000003
                Relay_Log_Pos: 266
        Relay_Master_Log_File: mysql-bin.000006
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB: platform
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
mysql.%,test.%,information_schema.%,performance_schema.%
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 875
              Relay_Log_Space: 1349
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 112
                  Master_UUID:
```

```
             Master_Info_File: /service/db/mysql/master.info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave
I/O thread to update it
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp:
               Master_SSL_Crl:
           Master_SSL_Crlpath:
           Retrieved_Gtid_Set:
            Executed_Gtid_Set:
                Auto_Position: 0
1 row in set (0.00 sec)
```

如果 Slave_IO_Running、Slave_SQL_Running的状态为No，则执行 start slave 启动 slave

在master、slave上可以分别使用 show processlist\G 查看I/O和SQL线程的运行情况；

```
# master
mysql> show processlist\G
*************************** 1. row ***************************
     Id: 10
   User: root
   Host: localhost
     db: repldb
Command: Query
   Time: 0
  State: NULL
   Info: show processlist
*************************** 2. row ***************************
     Id: 11
   User: repl
   Host: 112.74.68.188:37575
     db: NULL
Command: Binlog Dump
   Time: 5411
  State: Master has sent all binlog to slave; waiting for binlog to be updated
   Info: NULL
2 rows in set (0.00 sec)
```

```
# slave
mysql> show processlist\G
*************************** 1. row ***************************
     Id: 1
   User: system user
   Host:
     db: NULL
Command: Connect
   Time: 5454
  State: Waiting for master to send event
   Info: NULL
*************************** 2. row ***************************
     Id: 2
   User: system user
   Host:
     db: NULL
Command: Connect
```

```
   Time: 4897
  State: Slave has read all relay log; waiting for the slave I/O thread to update
it
   Info: NULL
*************************** 3. row ***************************
     Id: 3
   User: root
   Host: localhost
     db: repldb
Command: Query
   Time: 0
  State: init
   Info: show processlist
3 rows in set (0.00 sec)
```

3.4 从另一个master初始化slave

前面讨论的假设你是新安装的master和slave，所以，slave与master有相同的数据。但是，大多数情况却不是这样的，例如，你的master可能已经运行很久了，而你想对新安装的slave进行数据同步，甚至它没有master的数据。

此时，有几种方法可以使slave从另一个服务开始，例如，从master拷贝数据，从另一个slave克隆，从最近的备份开始一个slave。Slave与master同步时，需要三样东西：

(1)master的某个时刻的数据快照；

(2)master当前的日志文件、以及生成快照时的字节偏移。这两个值可以叫做日志文件坐标(log file coordinate)，因为它们确定了一个二进制日志的位置，你可以用SHOW MASTER STATUS命令找到日志文件的坐标；

(3)master的二进制日志文件。

可以通过以下几中方法来克隆一个slave：

(1)　冷拷贝(cold copy)

停止master，将master的文件拷贝到slave；然后重启master。缺点很明显。

(2)　热拷贝(warm copy)

如果你仅使用MyISAM表，你可以使用mysqlhotcopy拷贝，即使服务器正在运行。

(3)　使用mysqldump

使用mysqldump来得到一个数据快照可分为以下几步：

<1>锁表：如果你还没有锁表，你应该对表加锁，防止其它连接修改数据库，否则，你得到的数据可以是不一致的。如下：

mysql> FLUSH TABLES WITH READ LOCK;

<2>在另一个连接用mysqldump创建一个你想进行复制的数据库的转储：

shell> mysqldump --all-databases --lock-all-tables >dbdump.db

<3>对表释放锁。

mysql> UNLOCK TABLES;

三、实践

复制的用户名称：repl/30days

master：120.24.49.112（亿昇国际）

slave：112.74.68.188（自由城市）

四、问题

1、replicate-do-db/replicate-do-ignore-db与replicate_wild_do_table/replicate_wild_ignore_table 区别： 因为replicate-ignore-db 是通过use db来确定是否过滤的，对跨库操作的sql不会同步由主同步到从； 而wild-ignore是通过真实被修改的表进行过滤的，更为准确。

在使用replicate-do-db时如果从库同步出现错误（如，主库跨库建表然后切回到自身insert操作，从库同步insert找不到表），可以结合relay-log找到出错地方， SET GLOBAL SQL_SLAVE_SKIP_COUNTER = N; 跳过某些错误或者补充缺失的步骤。

参见： replicate_do_db和repalicate_wild_do_table参数设置 http://blog.itpub.net/23073137/viewspace-718693/

2、binlog_format=STATEMENT/ROW/MIXED 的区别：

（1）在binlog_format=STATEMENT时，在用use dbname的格式下，如果dbname没有在binlog-do-db里，DDL和DML语句都不会被记录在binlog里。即使指定具体的test.dd；

（2）在row模式下，在用use dbname的格式下，如果dbname没有在binlog-do-db里，DDL语句都不会被记录在binlog里。即使指定具体的test.dd；DML语句会记录。

（3）在mixed模式下，在用use dbname的格式下，如果dbname没有在binlog-do-db里，DDL、DML语句都不会被记录在binlog里。即使指定具体的test.dd；

参见：http://blog.sina.com.cn/s/blog_747f4c1d0102w9pp.html

五、扩展——常用的拓扑结构：

复制的体系结构有以下一些基本原则：

(1) 每个slave只能有一个master；

(2) 每个slave只能有一个唯一的服务器ID；

(3) 每个master可以有很多slave；

(4) 如果你设置log_slave_updates，slave可以是其它slave的master，从而扩散master的更新。

MySQL不支持多主服务器复制(Multimaster Replication)——即一个slave可以有多个master。但是，通过一些简单的组合，我们却可以建立灵活而强大的复制体系结构。

（1）单一主从

（2）主动模式的master-master（两台server互为master、slave，需要注意更新冲突带来的数据不一致问题）；

（3）主动被动模式的master-master（一台只能进行只读操作）

（4）带从服务器的master、master

http://www.cnblogs.com/ggjucheng/archive/2012/11/13/2768879.html

参考：http://www.cnblogs.com/hustcat/archive/2009/12/19/1627525.html

http://www.cnblogs.com/weafer/archive/2011/09/20/2182566.html