**MySQL数据库安装配置手册**

**(MySQL主主)**

|  |  |
| --- | --- |
| 文件版本 | V2.0 |
| 编写人 | 杨光、倪志荣 |
| 编写时间 | 2019年08月27日 |

**文件修改记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **修改日期** | **版本号** | **修改内容** | **修改人** | **审核人** | **批准人/日期** |
| 2018-08-28 | V1.0 | 初建文档，用于在SLES12 SP2操作系统环境下安装测试官网MySQL5.6.41数据库 | 姜文刚 |  |  |
| 2019-08-27 | V2.0 | 文档修订、补充 | 杨光、倪志荣 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目 录**

[1 引言 3](#_Toc17899279)

[1.1 文档用途 3](#_Toc17899280)

[1.2 MySQL主主方案概述 3](#_Toc17899281)

[2 CentOS操作系统安装 3](#_Toc17899282)

[2.1 CentOS操作系统安装 3](#_Toc17899283)

[2.2 操作系统参数配置 3](#_Toc17899284)

[3 磁盘规划 3](#_Toc17899285)

[4 所需软件及版本描述 4](#_Toc17899286)

[5 安装MySQL 4](#_Toc17899287)

[5.1 安装前配置 4](#_Toc17899288)

[5.2 安装 MySQL shell 5](#_Toc17899289)

[5.3 安装percona-toolkit 5](#_Toc17899290)

[5.4 MySQL安装 5](#_Toc17899291)

[5.5 MySQL初始化 8](#_Toc17899292)

[6 MySQL主主同步 8](#_Toc17899293)

[6.1 创建数据同步用户 8](#_Toc17899294)

[6.2 执行数据同步 8](#_Toc17899295)

[6.3 监控MySQL同步 9](#_Toc17899296)

[6.4 MySQL开发禁忌 9](#_Toc17899297)

[7 Keepalived部署 9](#_Toc17899298)

[8 MySQL备份与恢复 9](#_Toc17899299)

[9 MySQL DR方案 9](#_Toc17899300)

[10 MySQL日志及相关参数说明 9](#_Toc17899301)

[11 MySQL配置文件 9](#_Toc17899302)

[11.1 MySQL Master修改参数 9](#_Toc17899303)

[11.2 MySQL Master配置文件解释 9](#_Toc17899304)

[11.3 MySQL Slave配置文件解释 10](#_Toc17899305)

[12 MySQL巡检 10](#_Toc17899306)

[13 MySQL常用命令 10](#_Toc17899307)

[MySQL启停 10](#_Toc17899308)

[Show processlist 10](#_Toc17899309)

[Show master status和show slave status 10](#_Toc17899310)

[Show grant for <replication user> 10](#_Toc17899311)

[Change master 10](#_Toc17899312)

[Stop/Start salve 10](#_Toc17899313)

[Show Slave hosts 10](#_Toc17899314)

[Show binlog events; 10](#_Toc17899315)

[Mysqlbinlog 10](#_Toc17899316)

[Purge Binary logs 10](#_Toc17899317)

[Mysqldump 10](#_Toc17899318)

[Mysqlpdump 10](#_Toc17899319)

# 引言

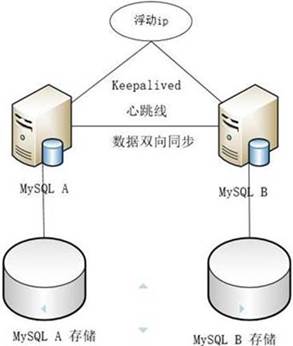
## 文档用途

本文档用于指导现场实施人员和项目维护人员完成以下工作：

* 指导进行操作系统安装
* 指导进行磁盘空间规划
* 进行MySQL数据库的安装
* 进行MySQL参数配置优化
* 实现MySQL的主主同步配置
* 进行数据库备份、DR方案实施
* 指导现场实施人员和项目维护人员进行数据库巡检及状态监控。

## MySQL主主方案概述

**方案架构如下：**



**方案描述：**

1. MySQL A与MySQL B同时启动，对数据的增删改操作实时双向同步，相同的数据存储两份，利用浮动IP对外提供服务。
2. MySQL A与MySQL B 数据文件独立存放在各自的存储上。
3. 当MySQL A对外提供服务时，浮动IP绑定在MySQL A上边，对A的操作会实时同步到B上。当MySQL A出现故障（例如，服务器宕机，MySQL程序停掉），Keepalived检测到后将浮动ip绑定到MySQL B上，B上的MySQL服务直接提供服务。

**方案优点：**

故障切换速度快

两套数据库独立运行，可靠性高。

**方案缺点：**

占用存储空间相对大。

两台MySQL做做主主同步消耗额系统资源。

需监控主从同步状态,防止主从同步异常;并在主从同步异常时,手动恢复主从同步.

# CentOS操作系统安装

## CentOS操作系统安装

CentOS操作系统镜像下载地址: [www.centos.org](http://www.centos.org).



**服务器信息列表：**

|  |  |  |
| --- | --- | --- |
| 主机名 | 服务器ip地址 | 部署数据库版本 |
| DB1 | 10.250.58.4 | MySQL-5.7.x |
| DB2 | 10.250.58.5 | MySQL-5.7.x |
| BACKUP | 10.250.58.6 | MySQL-5.7.x |
| VIP | 10.250.58.7 |  |

## 操作系统参数配置

网给方面的配置， 要修改/etc/sysctl.conf文件

#Swap调整(尽量不使用swap分区)

vm.swappiness=5

#增加tcp支持的队列数

net.ipv4.tcp\_max\_syn\_backlog = 65535

#减少断开连接时 ，资源回收

net.ipv4.tcp\_max\_tw\_buckets = 8000

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_fin\_timeout = 10

#句柄数

fs.file-max=65535

打开文件的限制

可以便用ulimit -a目录的刍各位限制，可以修改/etc/security/limits.conf文件 增加以下内容以修改打开文件数量的限制

\* soft nproc 65535

\* hard nproc 65535

\* soft nofile 65535

\* hard nofile 65535

除此之外最好在MySQL服务器上关闭iptables,selinux 等防火墙软件。

# 磁盘规划



建议优先使用SSD存储设备,以提高IO吞吐能力。

**注**:为分散IO,提高IO吞吐能力,redolog、binlog、relaylog单独VG，数据文件单独VG，备份文件单独VG，MySQL及其他软件存放本地磁盘。

|  |  |  |  |
| --- | --- | --- | --- |
| 目录名称 | 创建命令 | 大小 | 描述 |
| redolog | mkdir–p–m 755 /redolog | 150G以上 | 数据文件（logvg） |
| datadir | mkdir–p–m 755 /data | 业务数据量 | 数据文件（datavg） |
| binlog | mkdir–p–m 755 /binlog | 200G以上(与业务量相关) | 二进制日志（logvg） |
| soft | mkdir –p –m 755 /opt/mysql | 100G(系统本地磁盘) | 软件包（本地磁盘） |
| backup | mkdir –p -m 755 /backup | 数据量\*2.5以上 | 备份（backupvg） |

# 所需软件及版本描述

**MySQL软件包上传**

|  |  |
| --- | --- |
| 软件包名称 | 备注说明 |
| mysql-5.7.\*\*.el7.x86\_64.rpm-bundle.tar | 上传到/opt/soft/目录下 |
| Percona-XtraBackup-2.4.\*-el7-x86\_64-bundle.tar | 上传到/opt/soft/目录下 |
| percona-toolkit-3.0.\*-el7-x86\_64-bundle.tar | 上传到/opt/soft/目录下 |
| mysql-shell-\*\*\*.el7.x86\_64.rpm | 上传到/opt/soft/目录下 |
| CentOS-7-x86\_64-DVD-\*\*\*\*.iso | 上传到/opt/soft,并配置本地yum源 |

# 安装MySQL

## 安装前配置

**卸载操作系统自带的MySQL**

下面的操作在主机DB1和DB2的两台数据库服务器上操作。

查看系统是否默认安装MySQL。

# rpm -qa | grep mysql

perl-DBD-mysql-4.021-7.178.x86\_64

libmysqlclient18-10.0.27-12.1.x86\_64

通过命令查看到已安装的,非mysql开头的程序包可以不删除

通过如下命令删除：

# rpm -e perl-DBD-mysql-4.021-7.178.x86\_64 --nodeps

# rpm -e libmysqlclient18-10.0.27-12.1.x86\_64 --nodeps

**卸载操作系统自带的MariaDB数据库**

在两台数据库服务器上查看是否已经安装MariaDB数据库

# rpm -qa | grep -i MariaDB

mariadb-client-10.0.27-12.1.x86\_64

mariadb-errormessages-10.0.27-12.1.x86\_64

mariadb-10.0.27-12.1.x86\_64

卸载MariaDB数据库：

# rpm -e mariadb-10.0.27-12.1.x86\_64

# rpm -e mariadb-client-10.0.27-12.1.x86\_64

# rpm -e mariadb-errormessages-10.0.27-12.1.x86\_64

## 安装 MySQL shell

# cd /opt/soft/

# tar -xvf mysql-shell-\*\*\*.el7.x86\_64.rpm

# cd mysql\*

安装MySQL shell数据库：

# mkdir –p –m 755 /opt/mysql-shell

# rpm -ivh --prefix=/opt/mysql-shell \*.rpm

## 安装Percona Toolkit

# cd /opt/soft/

# tar -xvf percona-toolkit-3.0.\*-el7-x86\_64-bundle.tar

# cd mysql\*

安装percona toolkit数据库：

# mkdir –p–m 755 /opt/percona-toolkit

# rpm -ivh --prefix=/opt/percona-toolkit \*.rpm

Percona Toolkit 官方文档网址:

<https://www.percona.com/doc/percona-toolkit/LATEST/index.html>

附件为Percona Toolkit 3.0.13官方文档:





## MySQL安装

下面的操作在主机DB1和DB2两台数据库主机上。

进入程序包目录，解压软件包：

# cd /opt/soft/

# tar -xvf Mysql-5.7.\*\*.el7.x86\_64.rpm-bundle.tar

#cd mysql\*

安装MySQL数据库：

# mkdir –p –m 755 /opt/mysql

# rpm -ivh --prefix=/opt/mysql \*.rpm

## MySQL初始化

将my.cnf主配置文件拷贝到/etc/目录下，启动mysql服务

[client]

port = 3306

socket = /data/mysql.sock

[mysql]

prompt="\u@mysqldb \R:\m:\s [\d]> "

no-auto-rehash

[mysqld]

user = mysql

port = 3306

basedir = /opt/mysq

datadir = /data

socket = /data/mysql.sock

pid-file = mysqldb.pid

character-set-server = utf8mb4

skip\_name\_resolve = 1

#若你的MySQL数据库主要运行在境外，请务必根据实际情况调整本参数

default\_time\_zone = "+8:00"

open\_files\_limit = 65535

back\_log = 1024

max\_connections = 1500

max\_connect\_errors = 1000000

table\_open\_cache = 10000

table\_definition\_cache = 10000

table\_open\_cache\_instances = 64

thread\_stack = 512K

external-locking = FALSE

max\_allowed\_packet = 32M

sort\_buffer\_size = 4M

join\_buffer\_size = 4M

thread\_cache\_size = 2250

interactive\_timeout = 600

wait\_timeout = 600

tmp\_table\_size = 32M

max\_heap\_table\_size = 32M

slow\_query\_log = 1

log\_timestamps = SYSTEM

slow\_query\_log\_file = /binlog/slow.log

log-error = /data/error.log

long\_query\_time = 0.1

log\_queries\_not\_using\_indexes =1

log\_throttle\_queries\_not\_using\_indexes = 60

min\_examined\_row\_limit = 100

log\_slow\_admin\_statements = 1

log\_slow\_slave\_statements = 1

server-id = 3306

log-bin = /binlog

sync\_binlog = 1

binlog\_cache\_size = 4M

max\_binlog\_cache\_size = 2G

max\_binlog\_size = 1G

#注意：MySQL 8.0开始，binlog\_expire\_logs\_seconds选项也存在的话，会忽略expire\_logs\_days选项

expire\_logs\_days = 7

master\_info\_repository = TABLE

relay\_log\_info\_repository = TABLE

gtid\_mode = on

enforce\_gtid\_consistency = 1

log\_slave\_updates

slave-rows-search-algorithms = 'INDEX\_SCAN,HASH\_SCAN'

binlog\_format = row

binlog\_checksum = 1

relay\_log\_recovery = 1

relay-log-purge = 1

key\_buffer\_size = 32M

read\_buffer\_size = 8M

read\_rnd\_buffer\_size = 4M

bulk\_insert\_buffer\_size = 64M

myisam\_sort\_buffer\_size = 128M

myisam\_max\_sort\_file\_size = 10G

myisam\_repair\_threads = 1

lock\_wait\_timeout = 3600

explicit\_defaults\_for\_timestamp = 1

innodb\_thread\_concurrency = 0

innodb\_sync\_spin\_loops = 100

innodb\_spin\_wait\_delay = 30

transaction\_isolation = REPEATABLE-READ

#innodb\_additional\_mem\_pool\_size = 16M

innodb\_buffer\_pool\_size = 91750M

innodb\_buffer\_pool\_instances = 8

innodb\_buffer\_pool\_load\_at\_startup = 1

innodb\_buffer\_pool\_dump\_at\_shutdown = 1

innodb\_data\_file\_path = ibdata1:1G:autoextend

innodb\_flush\_log\_at\_trx\_commit = 1

innodb\_log\_buffer\_size = 32M

innodb\_log\_file\_size = 2G

innodb\_log\_files\_in\_group = 8

innodb\_log\_group\_home\_dir = /redolog

innodb\_max\_undo\_log\_size = 4G

innodb\_undo\_directory = /data/undolog

innodb\_undo\_tablespaces = 95

# 根据您的服务器IOPS能力适当调整

# 一般配普通SSD盘的话，可以调整到 10000 - 20000

# 配置高端PCIe SSD卡的话，则可以调整的更高，比如 50000 - 80000

innodb\_io\_capacity = 4000

innodb\_io\_capacity\_max = 8000

innodb\_flush\_sync = 0

innodb\_flush\_neighbors = 0

innodb\_write\_io\_threads = 8

innodb\_read\_io\_threads = 8

innodb\_purge\_threads = 4

innodb\_page\_cleaners = 4

innodb\_open\_files = 65535

innodb\_max\_dirty\_pages\_pct = 50

innodb\_flush\_method = O\_DIRECT

innodb\_lru\_scan\_depth = 4000

innodb\_checksum\_algorithm = crc32

innodb\_lock\_wait\_timeout = 10

innodb\_rollback\_on\_timeout = 1

innodb\_print\_all\_deadlocks = 1

innodb\_file\_per\_table = 1

innodb\_online\_alter\_log\_max\_size = 4G

innodb\_stats\_on\_metadata = 0

#注意：MySQL 8.0.16开始删除该选项

internal\_tmp\_disk\_storage\_engine = InnoDB

# some var for MySQL 5.7

innodb\_checksums = 1

#innodb\_file\_format = Barracuda

#innodb\_file\_format\_max = Barracuda

query\_cache\_size = 0

query\_cache\_type = 0

innodb\_undo\_logs = 128

innodb\_status\_file = 1

#注意: 开启 innodb\_status\_output & innodb\_status\_output\_locks 后, 可能会导致log-error文件增长较快

innodb\_status\_output = 0

innodb\_status\_output\_locks = 0

#performance\_schema

performance\_schema = 1

performance\_schema\_instrument = '%memory%=on'

performance\_schema\_instrument = '%lock%=on'

#innodb monitor

innodb\_monitor\_enable="module\_innodb"

innodb\_monitor\_enable="module\_server"

innodb\_monitor\_enable="module\_dml"

innodb\_monitor\_enable="module\_ddl"

innodb\_monitor\_enable="module\_trx"

innodb\_monitor\_enable="module\_os"

innodb\_monitor\_enable="module\_purge"

innodb\_monitor\_enable="module\_log"

innodb\_monitor\_enable="module\_lock"

innodb\_monitor\_enable="module\_buffer"

innodb\_monitor\_enable="module\_index"

innodb\_monitor\_enable="module\_ibuf\_system"

innodb\_monitor\_enable="module\_buffer\_page"

innodb\_monitor\_enable="module\_adaptive\_hash"

[mysqldump]

quick

max\_allowed\_packet = 32M

启动MySQL

# systemctl start mysql

# MySQL主主同步

下面的操作在主机DB1和DB2的两台数据库主机上操作。

**注**：若数据库业务量较大,网络负载存在瓶颈, 可以考虑将业务网卡与主主同步网卡分离,分别使用不同的网卡。

## 创建数据同步用户

服务器列表：

|  |  |  |
| --- | --- | --- |
| 主机名 | 服务器ip地址 | 数据同步用户 |
| DB1 | 10.250.58.4 | repl@'10.250.58.5' |
| DB2 | 10.250.58.5 | repl@'10.250.58.4' |

分别在两台数据库上创建数据同步用户，登入MySQL

在DB1上执行：

mysql>grant replication slave on \*.\* to repl@'10.250.58.5' identified by '123456';

在DB2上执行：

mysql>grant replication slave on \*.\* to repl@'10.250.58.4' identified by '123456';

## 执行数据同步

在DB1上执行

mysql>change master to master\_host='10.250.58.5',master\_port=3306, master\_user='repl',master\_password='123456',master\_log\_file='mysql-bin.000056',master\_log\_pos=25094639;

其中master\_log\_file和master\_log\_pos红色部分的值是通过在DB2上执行

show master status 查找到对应的File 和position的值。

在DB2上执行

mysql> change master to master\_host='10.250.58.4', master\_port=3306, master\_user='repl',master\_password='123456',master\_log\_file='mysql-bin.000056',master\_log\_pos=25094639;

其中master\_log\_file和master\_log\_pos红色部分的值是通过在DB1上执行

show master status 查找到对应的File 和position的值。

DB1和DB2服务器启动从服务器线程

mysql>start slave;

## 监控MySQL同步

DB1和DB2查看同步线程和IO线程

mysql> show slave status\G

查看下面两项是否为YES

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

都为yes两台mysql数据库主主复制成功。

# Keepalived部署

安装keepavlived

# yum install keepalived

DB1修改/etc/keepalived/keepavlived.conf配置文件

! Configuration File for keepalived

global\_defs {

notification\_email {

ops@wangshibo.cn

tech@wangshibo.cn

}

notification\_email\_from ops@wangshibo.cn

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id MASTER-HA

}

vrrp\_script chk\_mysql\_port { #检测mysql服务是否在运行。有很多方式，比如进程，用脚本检测等等

script "/opt/chk\_mysql.sh" #这里通过脚本监测

interval 2 #脚本执行间隔，每2s检测一次

weight -5 #脚本结果导致的优先级变更,检测失败(脚本返回非0)则优先级 -5

fall 2 #检测连续2次失败才算确定是真失败。会用weight减少优先级（1-255之间）

rise 1 #检测1次成功就算成功。但不修改优先级

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0 #指定虚拟ip的网卡接口

mcast\_src\_ip 10.250.58.4

virtual\_router\_id 51 #路由器标识，MASTER和BACKUP必须是一致的

priority 101 #定义优先级，数字越大，优先级越高，在同一个vrrp\_instance下，MASTER的优先级必须大于BACKUP的优先级。这样MASTER故障恢复后，就可以将VIP资源再次抢回来

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.250.58.7

}

track\_script {

chk\_mysql\_port

}

}

DB2修改/etc/keepalived/keepavlived.conf配置文件

! Configuration File for keepalived

global\_defs {

notification\_email {

ops@wangshibo.cn

tech@wangshibo.cn

}

notification\_email\_from ops@wangshibo.cn

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id MASTER-HA

}

vrrp\_script chk\_mysql\_port {

script "/opt/chk\_mysql.sh"

interval 2

weight -5

fall 2

rise 1

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth0

mcast\_src\_ip 10.250.58.5

virtual\_router\_id 51

priority 99

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

10.250.58.7

}

track\_script {

chk\_mysql\_port

}

}

DB1、DB2 编写切换脚本。KeepAlived做心跳检测，如果Master的MySQL服务挂了(3306端口挂了),那么它就会选择自杀。Slave的KeepAlived通过心跳检测发现这个情况，就会将VIP的请求接管。

# vim /opt/chk\_mysql.sh

#!/bin/bash

counter=$(netstat -na|grep "LISTEN"|grep "3306"|wc -l)

if [ "${counter}" -eq 0 ]; then

    /etc/init.d/keepalived stop

fi

# chmod 755 /opt/chk\_mysql.sh

# systemctl enable keepalived

# systemctl start keepalived

DB1和DB2两台服务器都要授权允许root用户远程登录，用于在客户端登陆测试！

mysql> grant all on \*.\* to root@'10.250.58.%' identified by "1234567";

Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;

Query OK, 0 rows affected (0.01 sec)

# MySQL备份与恢复

## 安装Percona XtraBackup

# cd /opt/soft/

# tar -xvf Percona-XtraBackup-2.4.\*-el7-x86\_64-bundle.tar

# cd Percona-XtraBackup\*

安装Percona XtraBackup数据库：

# mkdir -p –m 755 /opt/percona-xtrabackup

# rpm -ivh --prefix=/opt/percona-xtrabackup \*.rpm

Percona XtraBackup官方文档:

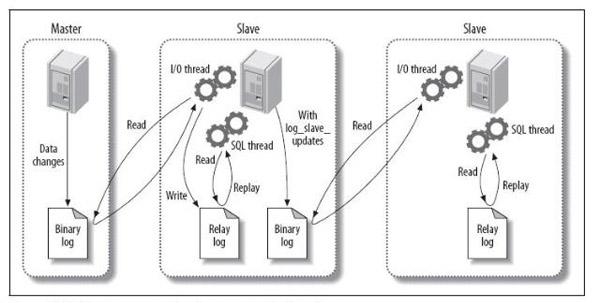
<https://www.percona.com/doc/percona-xtrabackup/LATEST/index.html>

## 数据库备份与恢复



# MySQL DR方案

## 方案架构



## 搭建主从同步

服务器列表：

|  |  |  |
| --- | --- | --- |
| 主机名 | 服务器ip地址 | 数据同步用户 |
| DB2 | 10.250.58.5 | repl@'10.250.58.6' |
| BACKUP | 10.250.58.6 |  |

分别在两台数据库上创建数据同步用户，登入MySQL

在DB2上执行：

mysql>grant replication slave on \*.\* to repl@'10.250.58.6' identified by '123456';

**执行数据同步:**

在BACKUP上执行

mysql> change master to master\_host='10.250.58.5', master\_port=3306, master\_user='repl',master\_password='123456',master\_log\_file='mysql-bin.000056',master\_log\_pos=25094639;

其中master\_log\_file和master\_log\_pos红色部分的值是通过在DB1上执行

show master status 查找到对应的File 和position的值。

BACKUP服务器启动从服务器线程

mysql>start slave;

# MySQL巡检



请保存巡检会话日志，并收集MySQL相关日志及操作系统日志；以方便查阅！

# MySQL日志及相关参数说明

MySQL 服务器上一共有六种日志：错误日志，查询日志，慢查询日志，二进制日志，事务日志，中继日志。

## 错误日志

1. 错误日志不仅仅记录错误信息，它记录的事件有：

* 服务器启动和关闭过程中的信息
* 服务器运行过程中的错误信息
* 事件调度器运行一个事件时产生的信息
* (如果被配置为从服务器)启动从服务器进程时产生的信息

1. 查看错误日志文件的路径

mysql> SHOW VARIABLES LIKE 'log\_error%';

1. 错误日志文件配置

在mysql数据库中，错误日志功能是默认开启的编辑 my.cnf（一般在mysql目录下），修改 log-error 参数,如果没有就新增：

# Error Logging.

log-error="filename.log"

相关配置变量说明

log\_error={1 | 0 | /PATH/TO/ERROR\_LOG\_FILENAME}

定义错误日志文件。作用范围为全局或会话级别，可用于配置文件，属非动态变量。

log\_warnings = {1|0}

决定是否将警告信息记录入错误日志。

## 查询日志

其中查询日志记录查询操作，默认情况下查询日志是关闭的。开启查询日志会增加很多磁盘 I/O， 所以如非出于调试目的，不建议开启查询日志。

查看查询日志是否启用及查询日志的路径

mysql> SHOW VARIABLES LIKE 'general\_log%';

配置查询日志

编辑 my.cnf，修改 general-log 参数为 1，同时设定 log-output 参数（日志输出类型）和 general\_log\_file 参数（查询日志路径）：

# General logging.：

log-output=FILE

general-log=1

general\_log\_file="filename.log"

保存 my.cnf 更改，重启 MySQL 服务。

## 慢查询日志

慢查询是指执行时长（包括等待CPU/IO的时间）超过 long\_query\_time 这个变量定义的时长的查询。慢查询日志开销比较小，可以用于定位性能问题，建议开启。

查看慢查询日志是否启用及慢查询日志的路径

mysql> SHOW VARIABLES LIKE 'slow\_query\_log%';

配置慢查询日志

编辑 my.cnf ，设置 log\_slow\_queries 参数为 1，同时设定 log-output 参数（日志输出类型）、slow-query-log\_file 参数（慢查询日志路径）和 long\_query\_time 参数：

# Slow logging.

log-output=FILE

log\_slow\_queries=1 //MySQL 5.6将此参数修改为了slow\_query\_log

slow\_query\_log\_file="filename.log"

long\_query\_time=10 //慢查的时长单位为秒，可以精确到小数点后6位(微秒)

保存 my.cnf 更改，重启 MySQL 服务。

关闭慢查询日志

设置 log\_slow\_queries 参数为 0：

# Slow logging.

log-output=NONE

log\_slow\_queries=0 //MySQL 5.6将此参数修改为了slow\_query\_log

slow\_query\_log\_file="filename.log"

long\_query\_time=10

保存 my.cnf 更改，重启 MySQL 服务。

## 二进制日志

二进制日志记录 MySQL 数据库中所有与更新相关的操作，即二进制日志记录了所有的 DDL（数据定义语言）语句和 DML（数据操纵语言）语句，但是不包括数据查询语句。常用于恢复数据库和主从复制。

查看 log\_bin 状态

mysql> SHOW VARIABLES LIKE 'log\_bin%';

启用二进制日志功能

编辑 my.cnf ，在 [mysqld] 下添加

# Binary Logging.

log-bin="filename-bin"

保存 my.cnf 更改，重启 MySQL 服务。

其他相关配置：

max\_binlog\_size={4096 .. 1073741824} ;

设定二进制日志文件上限，单位为字节，最小值为4K，最大值为1G，默认为1G。某事务所产生的日志信息只能写入一个二进制日志文件，因此，实际上的二进制日志文件可能大于这个指定的上限。作用范围为全局级别，可用于配置文件，属动态变量。

查看日志文件

在data目录下有一个mysql-bin.index便是索引文件，以mysql-bin开头并以数字结尾的文件为二进制日志文件。

查看所有的二进制文件：

mysql> show binary logs;

查看当前正在使用的二进制文件：

mysql> show master status;

二进制日志滚动

当 MySQL 服务进程启动、当前二进制日志文件的大小已经超过上限时、执行 FLUSH LOG 时，MySQL 会创建一个新的二进制日志文件。新的编号大1的日志用于记录最新的日志，而原日志名字不会被改变。

手动滚动命令：

mysql> flush logs;

查看日志详细

查看binlog日志有几种方式：

* 使用show binlog events方式可以获取当前以及指定binlog的日志，
* 使用mysqlbinlog命令行。

1. show binlog events方式

只查看第一个binlog文件的内容(show binlog events)

mysql> show binlog events; #默认会返回mysql-bin.000001的日志

查看指定binlog文件的内容(show binlog events in 'binname.xxxxx')

mysql> show binlog events in 'mysql-bin.000002';

获取指定位置binlog的内容(show binlog events from xxx)

mysql> show binlog events in 'mysql-bin.000002' from 107;

2. mysqlbinlog命令行

mysqlbinlog在mysql/bin目录

I 查看binlog日志

查看指定的binlog日志

[root@localhost bin]# ./mysqlbinlog ../data/mysql-bin.000003

按时间查看二进制日志

mysqlbinlog ../data/mysql-bin.000003 --start-datetime="2015-3-11 17:00:00"

mysqlbinlog ../data/mysql-bin.000003 --stop-datetime="2015-3-12 17:30:00"

mysqlbinlog ../data/mysql-bin.000003 --start-datetime="2015-3-11 17:00:00" --stop-datetime="2015-3-12 17:30:00"

按字节数查看二进制日志

mysqlbinlog ../data/mysql-bin.000003 --start-position=20

mysqlbinlog ../data/mysql-bin.000003 --stop-position=200

mysqlbinlog ../data/mysql-bin.000003 --start-position=20 --stop-position=200

过滤insert、update操作

mysqlbinlog ../data/mysql-bin.000003 | grep insert

II 查看binlog日志并输出

下面参考：使用mysqlbinlog提取二进制日志

c、提取指定position位置的binlog日志并输出到压缩文件

# mysqlbinlog --start-position="120" --stop-position="332" /opt/data/APP01bin.000001 |gzip >extra\_01.sql.gz

d、提取指定position位置的binlog日志导入数据库

# mysqlbinlog --start-position="120" --stop-position="332" /opt/data/APP01bin.000001 | mysql -uroot -p

e、提取指定开始时间的binlog并输出到日志文件

# mysqlbinlog --start-datetime="2014-12-15 20:15:23" /opt/data/APP01bin.000002 --result-file=extra02.sql

f、提取指定位置的多个binlog日志文件

# mysqlbinlog --start-position="120" --stop-position="332" /opt/data/APP01bin.000001 /opt/data/APP01bin.000002|more

g、提取指定数据库binlog并转换字符集到UTF8

# mysqlbinlog --database=test --set-charset=utf8 /opt/data/APP01bin.000001 /opt/data/APP01bin.000002 >test.sql

h、远程提取日志，指定结束时间

# mysqlbinlog -urobin -p -h192.168.1.116 -P3306 --stop-datetime="2014-12-15 20:30:23" --read-from-remote-server mysql-bin.000033 |more

i、远程提取使用row格式的binlog日志并输出到本地文件

# mysqlbinlog -urobin -p -P3606 -h192.168.1.177 --read-from-remote-server -vv inst3606bin.000005 >row.sql

expire\_logs\_days 参数

在 my.cnf 中配置 expire\_logs\_days 参数指定二进制日志的有效天数，MySQL 会自动删除过期的二进制日志。expire\_logs\_days 设置在服务器启动或者 MySQL 切换二进制日志时生效，因此，如果二进制日志没有增长和切换，服务器不会清除老条目。

# 二进制日志的有效天数

expire\_logs\_days = 5

清除二进制日志

清除所有日志（不存在主从复制关系）

mysql> RESET MASTER;

清除指定日志之前的所有日志

mysql> PURGE MASTER LOGS TO 'mysql-bin.000003';

清除某一时间点前的所有日志

mysql> PURGE MASTER LOGS BEFORE '2015-01-01 00:00:00';

清除 n 天前的所有日志

mysql> PURGE MASTER LOGS BEFORE CURRENT\_DATE - INTERVAL 10 DAY;

**警告**:由于二进制日志的重要性,请仅在确定不再需要将要被删除的二进制文件，或者在已经对二进制日志文件进行归档备份，或者已经进行数据库备份的情况下，才进行删除操作，且不要使用 rm 命令删除。

## 事务日志

出于性能和故障恢复的考虑，MySQL 服务器不会立即执行事务，而是先将事务记录在日志里面，这样可以将随机IO转换成顺序IO，从而提高IO性能。

事物日志通常是一组至少两个固定大小的文件，当其中一个写满时，MySQL会将事务日志写入另一个日志文件(先清空原有内容)。当 MySQL 从崩溃中恢复时，会读取事务日志，将其中已经 commit 的事务写入数据库，没有 commit 的事务 rollback 。

事务日志由存储引擎(innodb)管理，一般不需要手动干预。

## 中继日志

中继日志用于主从复制架构中的从服务器上，从服务器的 slave 进程从主服务器处获取二进制日志的内容并写入中继日志，然后由 IO 进程读取并执行中继日志中的语句。

## 日志分析工具

**mysqldumpslowmysql**：

官方提供的慢查询日志分析工具