## 案例七：监控MySQL主从状态

在生产环境里，常见的MySQL架构使用最多的就是主从了，所以对于主从是否同步数据的监控尤为重要。如果使用了专业的监控软件（如，zabbix）监控MySQL，那么选择监控工具提供的模板或插件去监控非常方便，但如果涉及到一些特殊要求就另当别论了。当然我们可以写shell脚本来实现定制化的需求。本案例需要写一个shell脚本来监控MySQL主从，具体要求如下：

1）每分钟检测一次，本次执行脚本时要检测上一次是否执行完成，如果还未完成则本次不执行

2）如果不同步需要发送告警邮件给admin@admin.com

3）需要做告警收敛，在没有解决问题之前需要每隔30分钟发一次告警邮件

4）假设本机mysql root账户密码为tpH40Kznv

知识点一：特殊符号$?

在命令行下或者在shell中，每执行完一条命令，都会有一个返回值，这个返回值为0表示该命令执行成功，非0表示执行不成功，返回值可以通过$?查看，如下所示：

# ls /etc/passwd

/etc/passwd

# echo $?

0

# ls /nofile

ls: 无法访问/nofile: 没有那个文件或目录

# echo $?

2

在shell脚本中，要想判断一条命令是否执行成功，就可以使用$?的值是否是0来实现。

知识点二：查看MySQL主从状态

判断MySQL主从是否正常有两种方法，第一种自然是对比两台MySQL上的数据是否一致，第二种是通过执行"show slave status"指令查看输出的结果，判断主从状态。对于第一种是最精准的，但是比较麻烦，所以我们通常用第二种方法来判断MySQL主从状态。操作如下所示（在从机上执行）：

# mysql -uroot -ptpH40Kznv -e "show slave status\G"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Slave\_IO\_State:

Master\_Host: 192.168.0.100

Master\_User: slave

Master\_Port: 3306

Connect\_Retry: 60

Master\_Log\_File: mysql-bin.010728

Read\_Master\_Log\_Pos: 1054294060

Relay\_Log\_File: db6-relay-bin.003759

Relay\_Log\_Pos: 283

Relay\_Master\_Log\_File: mysql-bin.010728

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

Replicate\_Do\_DB:

Replicate\_Ignore\_DB:

Replicate\_Do\_Table:

Replicate\_Ignore\_Table:

Replicate\_Wild\_Do\_Table:

Replicate\_Wild\_Ignore\_Table:

Last\_Errno:

Last\_Error:

Skip\_Counter: 0

Exec\_Master\_Log\_Pos: 937229962

Relay\_Log\_Space: 114889856

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:

Last\_IO\_Error:

Last\_SQL\_Errno:

Last\_SQL\_Error:

Replicate\_Ignore\_Server\_Ids:

Master\_Server\_Id: 3306

Master\_UUID: 9f911ef6-fbf5-11e4-b13c-b083feded312

Master\_Info\_File: /data/mysql/master.info

SQL\_Delay: 0

SQL\_Remaining\_Delay: NULL

Slave\_SQL\_Running\_State:

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

我们需要关注的行有：Slave\_IO\_Running、Slave\_SQL\_Running，只有这两行的值全都是Yes才算是主从同步状态正常。而任何一个为No，则不正常，当不正常时需要查看下面的Error（如，Last\_error、Last\_IO\_Error、Last\_SQL\_Error）相关信息近一步判断造成不同步的原因。

在生产环境中，有很大一部分问题不是主从不同步，而是主从延迟太严重，对于以上的输出信息中，有一行显示的是主从延迟信息：Seconds\_Behind\_Master，这个显示的是一个时间，单位是秒，表示从落后主多少秒。这个值其实并不完全精准，但我们却可以通过这个数值判断出主从是否有延迟（保证主从两台机器时间设置一致）。

知识点三：shell中的数学运算

这个知识点其实已经在前面案例中出现过，在这里我再做一个总结。shell中的加、减、乘、除和取余有点特殊，主要是格式上和C不太一样。

# a=1;b=2

# c=$[$a+$b] //加法

# echo $c

3

# d=$[$b-$a] //减法

# echo $d

1

# e=$[$a\*$b] //乘法

# echo $e

2

# f=$[$b/$a] //除法

# echo $f

2

# g=$[$b%$a] //取余

# echo $g

0

在shell的数学运算中要想使用小数，或者限定小数位数，可以借助bc的scale实现：

# a=10;b=3

# echo $[$a/$b]

3

# echo "scale=2;$a/$b"|bc //scale等于2表示小数有两位

3.33

知识点四：查看MySQL队列

在Linux系统里，我们可以通过ps、top等指令来查看系统的进程情况，而在MySQL中，可以通过"show processlist"指令查看MySQL的进程队列。当MySQL服务器负载变高或者访问卡顿时，查看一下进程队列是非常有必要的。操作如下：

# mysql -uroot -ptpH40Kznv -e "show processlist"

+-------+------+-----------+------+------------+------+-------+------------------+

| Id | User | Host | db | Command | Time | State | Info |

+-------+------+-----------+------+------------+------+-------+------------------+

| 24069 | root | localhost | NULL | Query | 0 | NULL | show processlist |

+-------+------+-----------+-------+-----------+------+-------+-------------------+

因为我实验的机器并没有什么访问量，所以并没有列出什么信息来，只有show processlist本身。在这里需要提醒大家，有时候列出来的队列命令太长从而显示不完整，则需要使用另外一个指令：show full processlist。

知识点五：shell中的内置变量

在shell脚本中，你会不会奇怪，哪里来的$1和$2，这其实就是shell脚本的预设变量，其中$1的值就是在执行的时候输入的第一个参数，而$2的值就是执行的时候输入的第二个参数，当然一个shell脚本的预设变量是没有限制的。另外还有一个$0，不过它代表的是脚本本身的名字。编写一个测试脚本，如下所示:

# vim option.sh //内容如下

#!/bin/bash

echo "$1 $2 $0"

执行结果如下：

# sh option.sh 1 2

1 2 option.sh

如果这样执行脚本，那$0的值会有所不同，如下：

# chmod a+x option.sh

# ./option.sh 1 2

1 2 ./option.sh

也就是说，$0的值会跟着执行时的命令变化，你用绝对路径它就会显示绝对路径，用相对路径它会显示相对路径。

本案例参考脚本：

要完成本脚本最核心的一个知识点就是查看slave状态，然后借助grep、sed、awk把我们想要的字符串截取出来，然后进行判断是否和预期一致。下面是我写的一个参考脚本，供大家揣摩学习，其中脚本中的mail.py需要参考案例二中知识点五：

#!/bin/bash

#检测MySQL主从是否同步

#作者：

#日期：2018-09-29

#版本：v0.2

#把脚本名字存入变量s\_name

s\_name=`echo $0|awk -F '/' '{print $NF}'`

Mysql\_c="mysql -uroot -ptpH40Kznv"

#该函数实现邮件告警收敛，在案例五种出现过，通用

m\_mail() {

log=$1

t\_s=`date +%s`

t\_s2=`date -d "1 hours ago" +%s`

if [ ! -f /tmp/$log ]

then

#创建$log文件

touch /tmp/$log

#增加a权限，只允许追加内容，不允许更改或删除

chattr +a /tmp/$log

#第一次告警，可以直接写入1小时以前的时间戳

echo $t\_s2 >> /tmp/$log

fi

#无论$log文件是否是刚刚创建，都需要查看最后一行的时间戳

t\_s2=`tail -1 /tmp/$log|awk '{print $1}'`

#取出最后一行即上次告警的时间戳后，立即写入当前的时间戳

echo $t\_s>>/tmp/$log

#取两次时间戳差值

v=$[$t\_s-$t\_s2]

#如果差值超过1800，立即发邮件

if [ $v -gt 1800 ]

then

#发邮件，其中$2为mail函数的第二个参数，这里为一个文件

python mail.py $mail\_user $1 "`cat $2`" 2>/dev/null

#定义计数器临时文件，并写入0

echo "0" > /tmp/$log.count

else

#如果计数器临时文件不存在，需要创建并写入0

if [ ! -f /tmp/$log.count ]

then

echo "0" > /tmp/$log.count

fi

nu=`cat /tmp/$log.count`

#30分钟内每发生1次告警，计数器加1

nu2=$[$nu+1]

echo $nu2>/tmp/$log.count

#当告警次数超过30次，需要再次发邮件

if [ $nu2 -gt 30 ]

then

python mail.py $mail\_user "$1 30 min" "`cat $2`" 2>/dev/null

#第二次告警后，将计数器再次从0开始

echo "0" > /tmp/$log.count

fi

fi

}

#把进程情况存入临时文件，如果加管道求行数会有问题

ps aux |grep "$s\_name" |grep -vE "$$|grep">/tmp/ps.tmp

p\_n=`wc -l /tmp/ps.tmp|awk '{print $1}'`

#当进程数大于0，则说明上次的脚本还未执行完

if [ $p\_n -gt 0 ]

then

exit

fi

#先执行一条执行show processlist，看是否执行成功

$Mysql\_c -e "show processlist" >/tmp/mysql\_pro.log 2>/tmp/mysql\_log.err

#如果上一条命令执行不成功，说明这个MySQL服务出现了问题

if [ $? -gt 0 ]

then

m\_mail mysql\_service\_error /tmp/mysql\_log.err

exit

else

$Mysql\_c -e "show slave status\G" >/tmp/mysql\_s.log

n1=`wc -l /tmp/mysql\_s.log|awk '{print $1}'`

if [ $n1 -gt 0 ]

then

y1=`grep 'Slave\_IO\_Running:' /tmp/mysql\_s.log|awk -F : '{print $2}'|sed 's/ //g'`

y2=`grep 'Slave\_SQL\_Running:' /tmp/mysql\_s.log|awk -F : '{print $2}'|sed 's/ //g'`

if [ $y1 == "No" ] || [ $y2 == "No" ]

then

m\_mail mysql\_slavestatus\_error /tmp/mysql\_s.log

fi

fi

fi