[MySQL FAQ]系列 — MySQL复制中slave延迟监控

在MySQL复制环境中，我们通常只根据 **Seconds_Behind_Master** 的值来判断SLAVE的延迟。这么做大部分情况下尚可接受，但并不够准确，而应该考虑更多因素。

首先，我们先看下SLAVE的状态：

```
yejr@imysql.com [(none)]> show slave status\G
*************************** 1. row ***************************
Slave_IO_State: Waiting for master to send event
***
Master_Log_File: mysql-bin.000327
Read_Master_Log_Pos: 668711237
Relay_Log_File: mysql-relay-bin.002999
Relay_Log_Pos: 214736858
Relay_Master_Log_File: mysql-bin.000327
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
***
Skip_Counter: 0
Exec_Master_Log_Pos: 654409041
Relay_Log_Space: 229039311
***
Seconds_Behind_Master: 3296
***
```

可以看到 **Seconds_Behind_Master** 的值是 **3296**，也就是SLAVE至少延迟了 **3296** 秒。

我们再来看下SLAVE上的2个REPLICATION进程状态：

```
yejr@imysql.com [(none)]> show full processlist\G
*************************** 1. row ***************************
Id: 6
User: system user
Host:
db: NULL
Command: Connect
Time: 22005006
State: Waiting for master to send event
Info: NULL
*************************** 2. row ***************************
Id: 7
User: system user
Host:
db: NULL
Command: Connect
Time: 3293
State: Updating
Info: UPDATE ** SET ** WHERE **
```

可以看到SQL线程一直在执行UPDATE操作，注意到 Time 的值是 3293，看起来像是这个UPDATE操作执行了3293秒，一个普通的SQL而已，肯定不至于需要这么久。

实际上，在REPLICATION进程中，Time 这列的值可能有几种情况：

1、SQL线程当前执行的binlog（实际上是relay log）中的timestamp和IO线程最新的timestamp的差值，这就是通常大家认为的 Seconds_Behind_Master 值，并不是某个SQL的实际执行耗时；

2、SQL线程当前如果没有活跃SQL在执行的话，Time值就是SQL线程的idle time；

而IO线程的Time值则是该线程自从启动以来的总时长（多少秒），如果系统时间在IO线程启动后发生修改的话，可能会导致该Time值异常，比如变成负

数，或者非常大。

来看下面几个状态：

```
#设置pager，只查看关注的几个status值
yejr@imysql.com [(none)]> pager cat | egrep -i 'system user|Exec_Master_Log_Pos|Seconds_Behind_Master|Read_Master_Log_Pos'

#这是没有活跃SQL的情况，Time值是idle time，并且 Seconds_Behind_Master 为 0
yejr@imysql.com [(none)]> show processlist; show slave status\G
| 6 | system user | | NULL | Connect | 22004245 | Waiting for master to send event | NULL |
| 7 | system user | | NULL | Connect | 13 | Has read all relay log;**
Read_Master_Log_Pos: 445167889
Exec_Master_Log_Pos: 445167889
Seconds_Behind_Master: 0

#和上面一样
yejr@imysql.com [(none)]> show processlist; show slave status\G
| 6 | system user | | NULL | Connect | 22004248 | Waiting for master to send event | NULL |
| 7 | system user | | NULL | Connect | 16 | Has read all relay log;**
Read_Master_Log_Pos: 445167889
Exec_Master_Log_Pos: 445167889
Seconds_Behind_Master: 0

#这时有活跃SQL了，Time值是和 Seconds_Behind_Master 一样，即SQL线程比IO线程“慢”了1秒
yejr@imysql.com [(none)]> show processlist; show slave status\G
| 6 | system user | | NULL | Connect | 22004252 | Waiting for master to send event | NULL |
| 7 | system user | | floweradmin | Connect | 1 | Updating | update **
Read_Master_Log_Pos: 445182239
Exec_Master_Log_Pos: 445175263
Seconds_Behind_Master: 1

#和上面一样
yejr@imysql.com [(none)]> show processlist; show slave status\G
| 6 | system user | | NULL | Connect | 22004254 | Waiting for master to send event | NULL |
| 7 | system user | | floweradmin | Connect | 1 | Updating | update **
Read_Master_Log_Pos: 445207174
Exec_Master_Log_Pos: 445196837
Seconds_Behind_Master: 1
```

好了，最后我们说下如何正确判断SLAVE的延迟情况：

1、首先看 **Relay_Master_Log_File** 和 **Master_Log_File** 是否有差异；

2、如果**Relay_Master_Log_File** 和 **Master_Log_File** 是一样的话，再来看**Exec_Master_Log_Pos** 和 **Read_Master_Log_Pos** 的差异，对比SQL线程比IO线程慢了多少个binlog事件；

3、如果**Relay_Master_Log_File** 和 **Master_Log_File** 不一样，那说明延迟可能较大，需要从MASTER上取得binlog status，判断当前的binlog和MASTER上的差距；

因此，相对更加严谨的做法是：

在第三方监控节点上，对MASTER和SLAVE同时发起**SHOW BINARY LOGS**和**SHOW SLAVE STATUS\G**的请求，最后判断二者binlog的差异，以及**Exec_Master_Log_Pos** 和 **Read_Master_Log_Pos** 的差异。

例如：

在MASTER上执行**SHOW BINARY LOGS** 的结果是：

```
+------------------+--------------+
| Log_name | File_size |
+------------------+--------------+
| mysql-bin.000009 | 1073742063 |
| mysql-bin.000010 | 107374193 |
+------------------+--------------+
```

而在SLAVE上执行**SHOW SLAVE STATUS\G** 的结果是：

```
Master_Log_File: mysql-bin.000009
 Read_Master_Log_Pos: 668711237
Relay_Master_Log_File: mysql-bin.000009
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
***
Exec_Master_Log_Pos: 654409041

***
Seconds_Behind_Master: 3296
***
```

这时候，SLAVE实际的延迟应该是：
mysql-bin.000009 这个binlog中的binlog position **1073742063** 和 SLAVE上读取到的binlog position之间的差异延迟，即：

```
1073742063 - 654409041 = 419333022 个binlog event
```

并且还要加上 **mysql-bin.000010**这个binlog已经产生的**107374193**个binlog event，共

```
107374193 + 419333022 = 526707215 个binlog event
```

后记更新：

1、可以在MASTER上维护一个监控表，它只有一个字段，存储这最新时间戳（高版本可以采用event_scheduler来更新，低版本可以用cron结合自动循环脚本来更新），在SLAVE上读取该字段的时间，只要MASTER和SLAVE的系统时间一致，即可快速知道SLAVE和MASTER延迟差了多少。不过，在高并发的系统下，这个时间戳可以细化到毫秒，否则哪怕时间一致，也是有可能会延迟数千个binlog event的。
2、网友（李大玉，QQ：407361231）细心指出上面的计算延迟有误，应该是mysql-bin.000009的最大事件数减去已经被执行完的事件数，即1073742063 – 654409041= 419333022个binlog event，再加上mysql-bin.000010这个binlog已经产生的107374193个binlog event，共526707215 个binlog event。

轩脉刃
2014/09/2210:43 上午

维护一个监控表的哪个方法很有意思哇

回复↓
