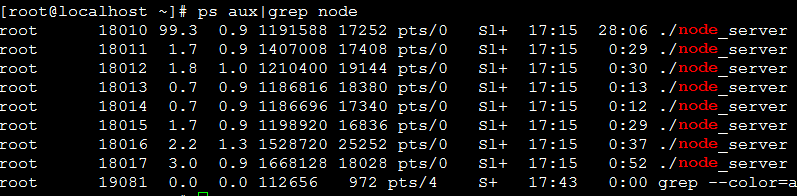
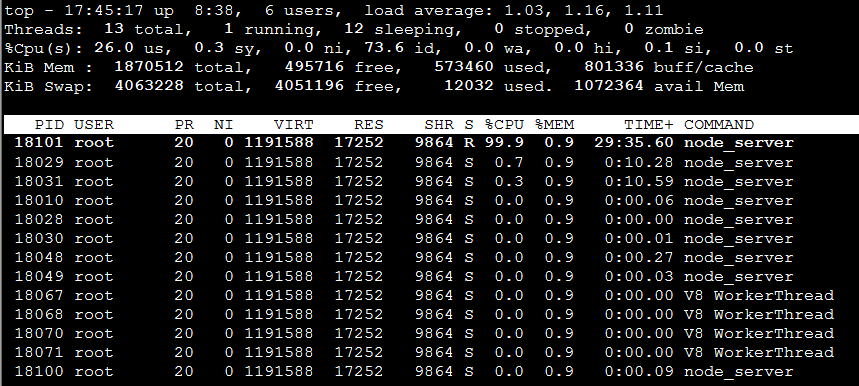
**服务器问题及解决方案**

1. Center\_Server进程死循环

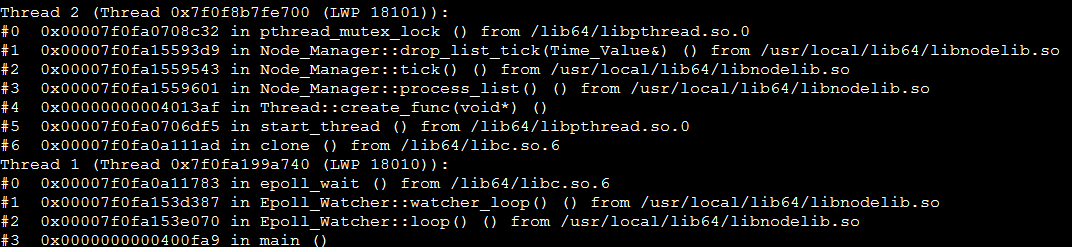
1、使用ps命令查看进程状态，发现center\_server进程cpu占用过高，进程id 18010

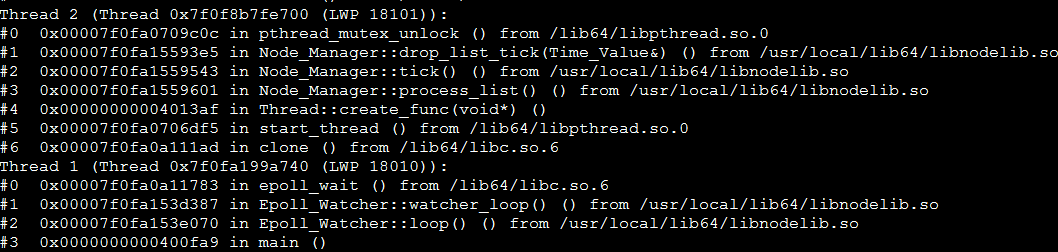


2、使用top –H –p 18010命令查看该进程的线程状态，发现占用cpu过高的是18101线程

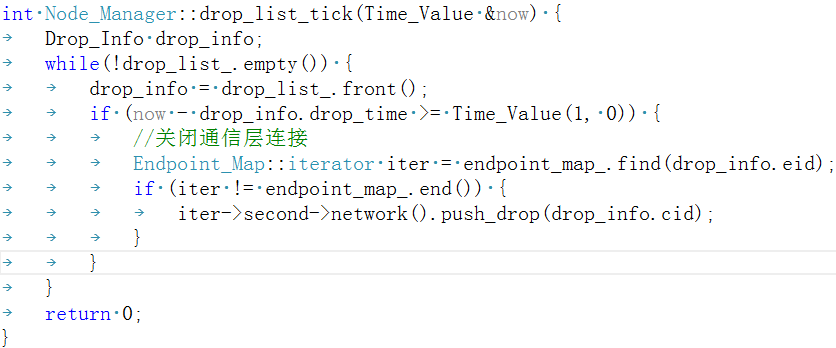


3、使用pstack 18010查看进程运行堆栈，发现是drop\_list\_tick出问题

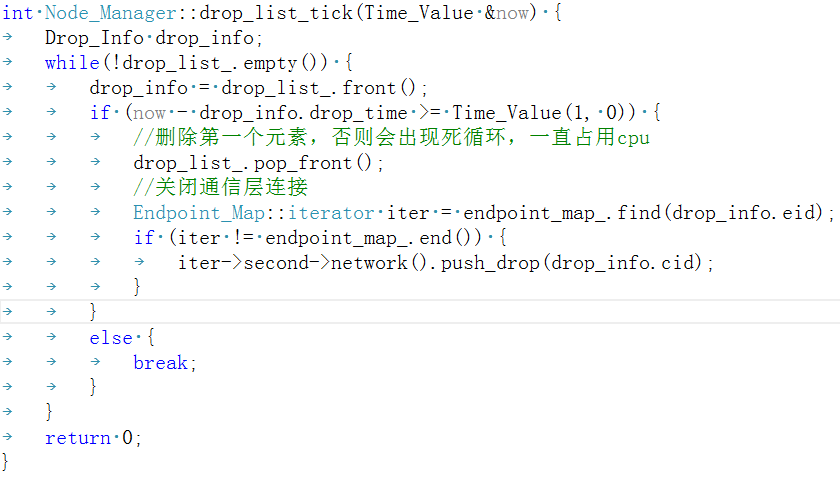




4、查看drop\_list\_tick,发现drop\_list忘记pop\_front，导致一直死循环



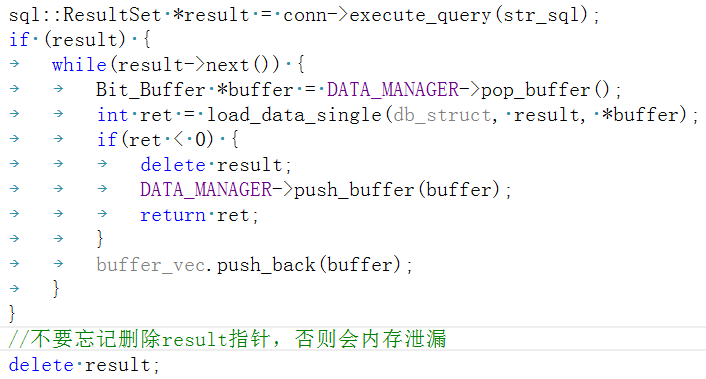
修改后如下：



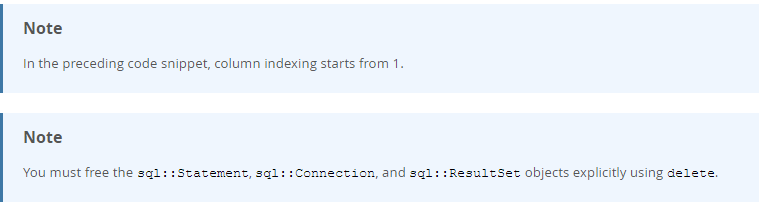
1. Data\_Server内存泄漏

跑大量机器人时候发现，data\_server内存一直在增长，看了机器人发的消息，里面有个创建公会消息，该消息会不断的发，每次创建公会时候根据公会名称查询公会是否存在，该操作直接查询数据库，去掉该消息后发现内存不再增长，截图如下：

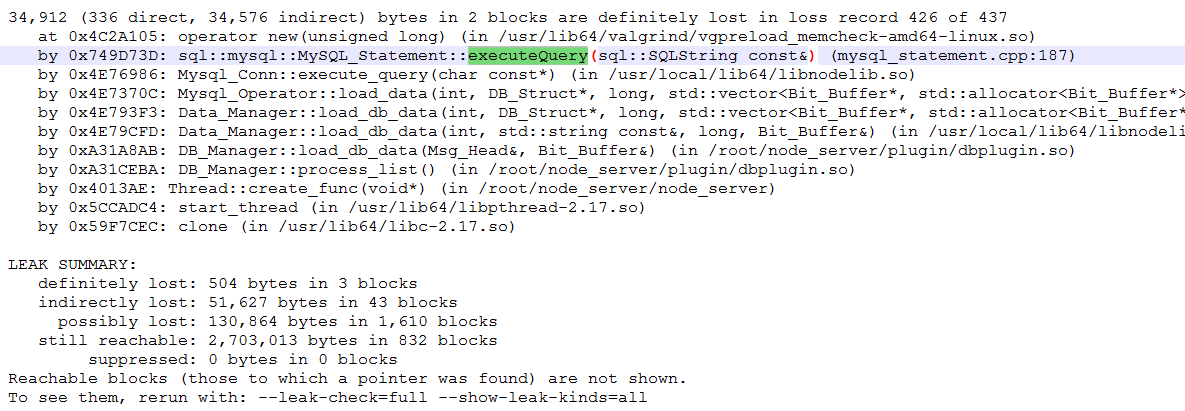
查看该函数代码，发现函数内容很简单，就是查询数据库，返回一个结果集，然后将结果集内容写入buffer,逻辑上不会有问题，如果出问题，肯定是结果集出问题了，通过百度发现，该结果集在使用后必须自己delete，否则会出现内容泄漏，加上delete就好了，代码如下：



可以参考<http://dev.mysql.com/doc/connector-cpp/en/connector-cpp-examples-results.html>

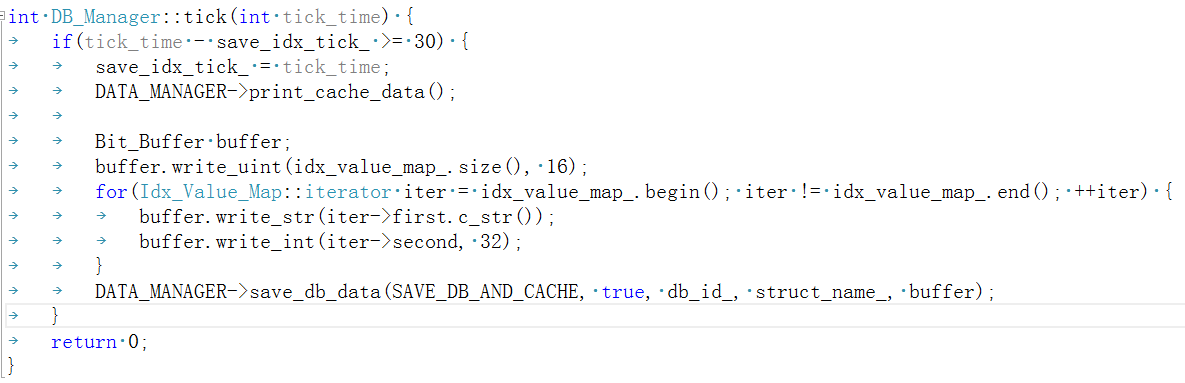


使用valgrind监控Data\_Server，测试报告如下：



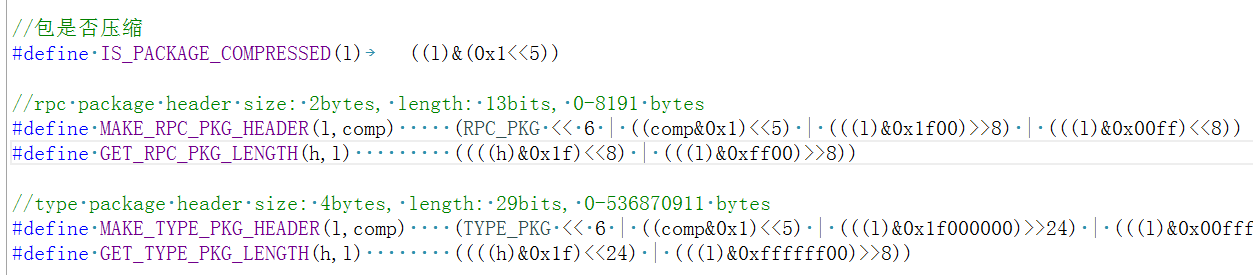
三、Data\_Server内存泄漏

大量机器人运行时候，每隔一段时间会core一次，gdb调试core，发现core在DB\_Manager的tick函数里面，该函数代码如下：

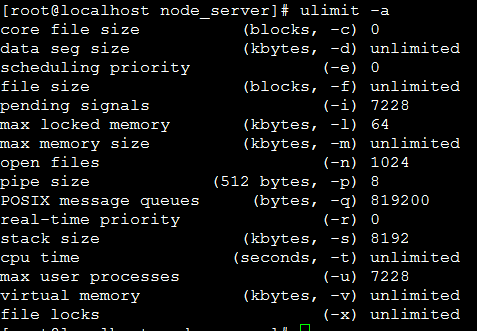
该函数会定时保存idx表，看了下函数调用堆栈，原来是在v8线程tick到期时候调用该函数，由于此处没有对idx\_value\_map\_加锁，新机器人登陆时候会创建id，使用该map,肯定会出错，于是在该map使用前加锁，使用后解锁，重新测试，测试一段时间后发现还是会core，而且core在mysql库函数调用处，思考了下，由于mysql连接指针是在db线程里面创建，v8线程直接调用保存数据时候，也有可能出错，于是改了实现机制，v8线程tick到期时候将tick时间放到db线程，由db线程自己循环tick处理，这样就不会由于多线程问题导致报错了。

四、公会表有大量数据存在无法正确加载的问题

public\_server进程在启动时候，会将所有公会数据加载到内存，公会数量少的时候，可以正确加载，当公会数据量大的时候，发现公会数据无法正确加载，buffer会报错，通过打印加载日志发现，data\_server进程加载数据和发送数据都是正确的，当消息buffer到了public\_server时候，包长度错了，比正确的包长度少了好多字节，查找包长加密解密函数发现，RPC消息包长只用了13位，最多只能8191个字节，当包长度超过这个限制时候，便会报错。解决办法是当包长度超过8191个字节时候，将消息类型改为Type\_Pkg,最大包长可以支持29位，可以解决问题。



五、Gate\_Server在高并发情况下卡死问题

使用机器人进行压力测试时候，发现当登陆2000个左右机器人时候，后面的机器人再也无法登陆，gate\_server进程cpu达到了100%左右，查看日志发现，在accept时候，返回-1，循环会一直continue，并写错误日志，导致cpu爆掉，此时错误号errno:24(EMFILE),表示系统打开的文件描述符太多，百度了下这个错误，发现是因为linux每个进程默认最大打开的描述符数量是1024，使用命令ulimit –a可以看到,解决办法是修改这个值即可，提高进程资源限制值。 

另外，由于写log功能集成在每个线程，导致log数量过多时候，会将逻辑包阻塞，导致包累积过多，将log功能放到单独的线程处理可解决问题。

六、Game\_Server高并发时无法正确添加player

使用机器人压力测试时候发现，game处理客户端消息时候，无法根据sid找到player，导致无法处理消息报错，查看client登陆逻辑发现，当gate验证session成功后，会同时通知game和client,此时game将session信息保存，client开始发登陆消息，由于进程间消息通信顺序无法保证，导致client消息发送到game时候，game可能还未正确添加session,无法根据sid找到player,就无法正确处理消息了。当server移除session时候，由于先player下线，再通知gate，导致client有新消息发来时候，player已经下线，但是gate未移除session，game就会报错。

解决办法：

gate增加session时候，先通知game，当game增加session完成后，发消息通知gate，此时gate再通知client建立session成功，可以保证正确的登陆顺序。

game移除session时候，先通知gate，当gate移除session完成后，发消息通知game，此时game再通知player下线，可以保证正确的下线顺序。另外当client直接断开与gate的网络层连接时候，gate需要先移除session,然后通知game让player下线。

七、Game\_Server高并发时内存泄漏

时间：2016-12-13 15:14:00，3700个机器人全部登陆，间隔500ms发送254和255



时间：2016-12-13 15:24:00，3700个机器人全部登陆，间隔500ms发送254和255

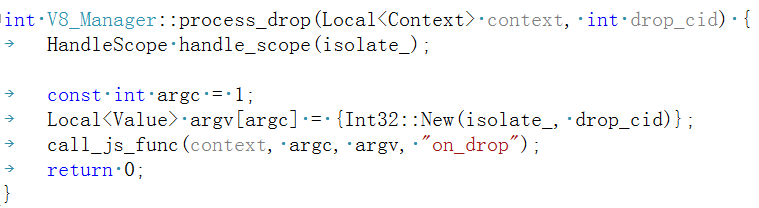


时间：2016-12-13 15:30:00，game\_server内存持续增长，接近崩溃



去掉定时发送254和255消息，game\_server内存不再增长

 通过以上四幅图对比分析可以发现，game\_server内存不断增长是由于不断发送254和255消息，v8对象没有及时清除，导致v8内存泄漏。查看V8\_Manager的process\_list函数发现，此函数在v8线程创建后就开始运行，在函数开始创建了一个handle\_scope，然后循环处理定时器，消息，掉线cid等，由于每次处理时候，都会在该handle\_scope内创建Local<Handle>,但是该handle\_scope不会释放，导致内存一直增长，无法被gc回收。将每个功能处理放到单独的函数内，每个函数创建自己的handle\_scope即可解决内存无限增长的问题。代码如下：



可以参考：

<http://blog.csdn.net/sunbxonline/article/details/20310897>

运行结果如下：

