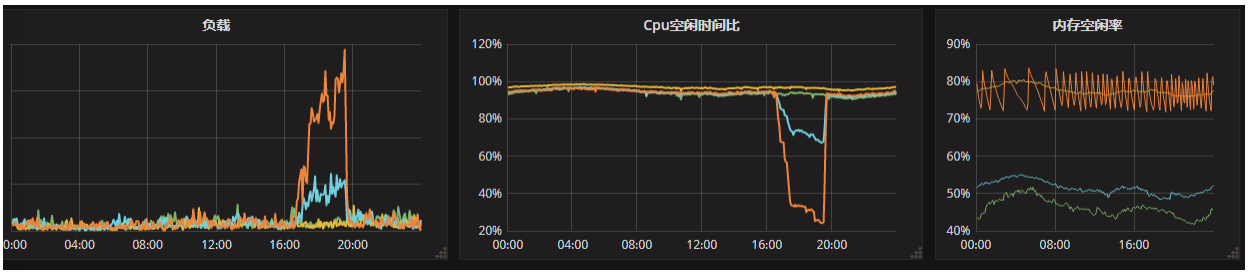
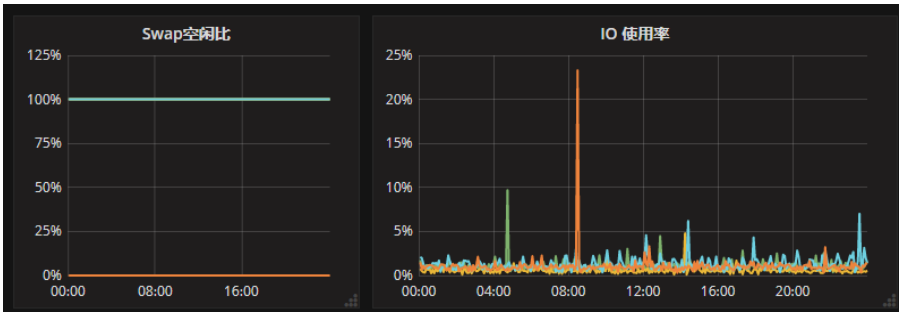
一、问题描述：

线上老用户可正常登陆，新用户注册登录失败，新用户多次点击后数据组件所在服务器的cpu空闲率从平时的90%降至30%，登录服务所在服务器的cpu空闲率从平时的90%降至70%





二、问题分析：

1. 从机器性能分析图中可看出，内存使用率和io使用率未发生异常波动，单cpu使用率上升，进而导致负载上升，（但因为负载不是特别高，所以服务器还能承受线上玩家的正常操作）

从内存和io的正常，排除内存泄露的问题，**cpu飙升**的话大概率就是**代码中有死循环**消耗了机器的cpu。（但是最近并没有更新代码，唯一操作有差异的就是使用了新手机注册游客账号失败，fb注册正常。）

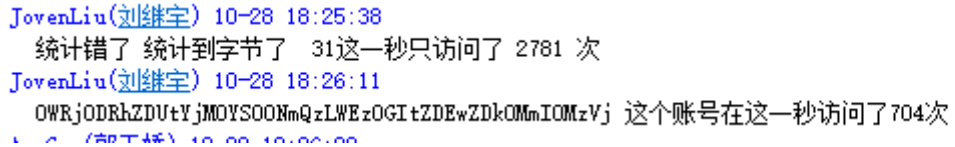
2. 查看两台异常的机器发现：dtsvr服务的cpu使用率180%，平时是5%左右，api-login服务的cpu使用率是20%，平时是2%左右，所以由此确定是登录服务出现异常。

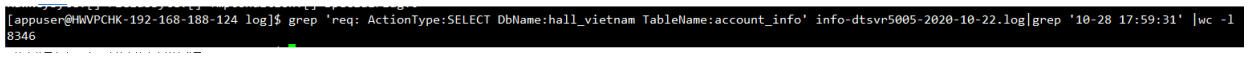
```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
13609 appuser   20   0  710452  78180   7584 S 186.3  0.5 403:16.89 dtsvrnew
17442 appuser   20   0  990944  72228   3968 S  75.0  0.4   3858:30 dtsrv
32607 appuser   20   0 1134452  92528   3812 S   4.3  0.6 104819:59 dtsrv
26366 root      20   0 2152068   1.9g  10548 S   3.7 12.5   2:12.27 all-in-one-linu
   24 root      20   0       0      0      0 S   2.7  0.0 429:39.82 ksoftirqd/3
```

3. 查看dtsvr，api-login两个服务的日志，api-login中无异常报错日志，dtsvr因为没有开启debug日志所以也看不到任何有效的日志。

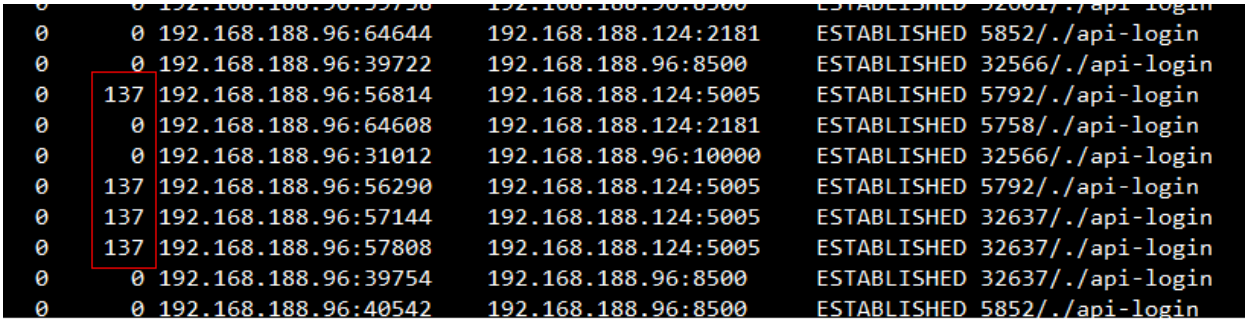
4. 让测试人员停止使用新手机注册游客登录，发现cpu使用率并没有下降，为了方便查问题，将dtsvr服务中开启的火焰图开启并分析，并未从中发现什么异常（也有可能是分析的不对）。最后将dtsvr服务的debug日志打开了一分钟后关闭，发现了问题：有非常多的异

常请求，即请求账户信息account_info这张表的操作相同的账号一秒查询请求了几百甚至上千多次数据库，但是在登录服务中该账号的登录请求只有个位数次。





5. 为了确定这个数据库查询操作时api-login服务登录请求后的死循环还是dtsvr服务中的死循环，我们在dtsvr这台服务器上抓包进行查看（tcpdump），发现请求来源是api-login服务所在的这台机器，那就确定问题了，应该是api-login服务的登录接口中有死循环。



使用netstat查看网络数据传输情况，登录发送给数据组件的请求数据来不及处理（侧面说明登录服务发送了很多请求给dtsvr服务）

6. 查看api-login.Login接口，这部分代码中并没有使用while,for等循环语句，但是有一个goto跳转语句，仔细看发现在某种情况下，确实存在goto不断执行同一块代码块的情况，即出现死循环，至此，发现并确定问题所在。

7. 问题代码展示：

```
if req.LoginType == entity.LOGIN_TYPE_VISITOR { //游客登录直接去账号表查找
    account, err = service.LoginDB.GetAccount(req.LoginType, req.Account)
    if err != nil {
        resp.Code = entity.CODE_ERR_SQL
        resp.Msg = "error get account by type"
        return nil
    }
    findSsaid := false
GoFind:
    if account.Uid > 0 {
        bindInfo, err = service.LoginDB.GetBindInfoByUid(account.Uid) //是否绑定了其他平台，有
        if err != nil {...}
        if bindInfo.Uid > 0 {...}
        if !findSsaid {...}
    } else {
        if req.Ssaid != "" {
            guid, _ := service.CheckDBService.GetGuid(req.Ssaid)
            if guid != "" {
                account, err = service.LoginDB.GetAccount(req.LoginType, guid)
                if err != nil {
                    plog.Warn("get account by ssaid err:", err)
                }
                resp.Data.Guid = guid
                findSsaid = true
                if account.Uid > 0 {
                    goto GoFind
                }
            } else {
                service.CheckDBService.SetGuid(req.Ssaid, req.Account)
            }
        }
    }
```


bug描述：

1. 新用户以游客身份登录，进入第一行代码，根据请求账号account和登录方式请求账户信息，因为是新用户，所以account为空，进入到下面的else分支，根据请求的ssaid设备id查找是否存在游客唯一guid,因为是新用户，guid也为空，继续走else分支，对guid行设置数据存储，然后走之后游客注册的逻辑（代码为展示），但因为每天游客注册限制原因，这一次注册失败。

2. 为注册成功后，再次在app中点击登录请求，再次进入该代码块中。因为注册限制导致上次注册没有注册成功，所以account信息依然为空，进入第一个else分支，但是这次根据ssaid查找guid是有值的，所欲进入 if guid != "" {}分支，根据guid和登录方式继续请求账户信息，因为是新用户，所以account继续为空，代码跳转togo GoFind，不断的根据guid和登录方式请求不存在的账户信息，继而进入死循环，导致出现异常。

总结：
代码还是写的少，代码逻辑不严谨不清晰，代码需要经常review
建议：
多看大佬的代码，学习代码结构，写高质量代码

调试确定代码中的死循环位置：
https://studygolang.com/articles/11881?fr=sidebar