# 试用期工作内容

## 熟悉代码

熟悉了解了接入层框架的设计、一些实现细节、还有接入层在整个业务流程中充当的角色。这有助于我以后在工作中更好地与各位同事进行交互，同时，也为接下来对接入层服务器偶发错误的解决以及对接入层服务器的性能优化打下了坚实的基础。

## 服务器压测

协助导师进行了服务器数据的压测：客户端分别在非安全或安全的状况下，向服务器发起足够多的连接数。

服务器的测试重点考察以下指标：a)一台服务器最大能接受多少连接数 b)服务器跟特定个(如五万连接数)客户端完成连接建立的时间

然后通过改变服务器的参数，包括worker线程数，io线程数、mysql\_pool大小、redis\_pool大小等，使服务器能接收的客户端数最大、接收特定数目客户端的时间最短。

经过多组数据的测量之后找出了相对来说的最优解，因此此次压测工作在一定程度上优化了服务器性能。

## 编写客户端

用go编写了一个支持建立多个连接的客户端，以替代之前的python客户端，获得更好的效率，更方便了以后的服务器测试。

* + 1. **为什么go的速度更快，有数据支撑吗？**
* **最简单的测试**

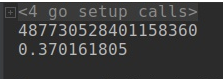
分别用python和go写了计算0到987654321累加和

python结果：



可以看出python运行运算花费了70多秒

go结果：



可以看出go只用了0.37秒。

运行时间单位统一都是秒，进行如此简单的运算结果差异都这么大，答案可想而知

* **测试多线程并发的结果**

可参考的网站：<https://studygolang.com/articles/1686>

利用go和python的各种框架实现多线程并发的web服务器作各个维度的对比，结果如下：

GOMAXPROCS=1, 一个 Python 进程：



GOMAXPROCS=2, 两个 Python 并发进程：



GOMAXPROCS=4, 四个 Python 并发进程：



可以看到，go几乎完胜了所有的用例，因此速度孰优孰劣无需多言。

* + 1. **为什么用go比python更好？**

速度更快意味着发起连接更快，消耗的系统资源也没这么多，可以比较稳定地发起五万以上的连接数，更加方便测试，也更加考验服务器的性能。

* + 1. **GOMAXPROCS和gorount的区别，客户端底层如何设计？数据如何加密？**

GOMAXPROCS相当于线程，即系统级别的核数；gorount是用户级的轻量级线程，即协程。

客户端底层封装了两个接口，一个是建立一个连接之后处理该连接的读写、关闭等。另一个接口管理所建立的所有连接。

使用crypto/aes包中的EncyptogAES函数和DecrptogAES函数进行加密和解密。

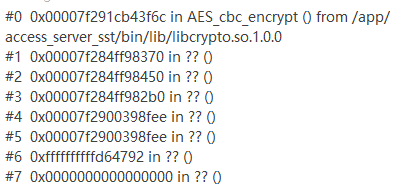
## 学习如何进行线上错误的发生以及错误排查

学习了当服务器coredump时如何用gdb结合core文件定位错误，查看错误处相关的堆栈以及值的变化，从而在源代码上作出相应修改，解决错误。

## 协作导师对线上问题的发生进行解决

协助导师解决线上服务器发生的问题，提供自己的思路，让导师进行多种角度的思考，从而更好地解决问题。协助解决的问题如下：

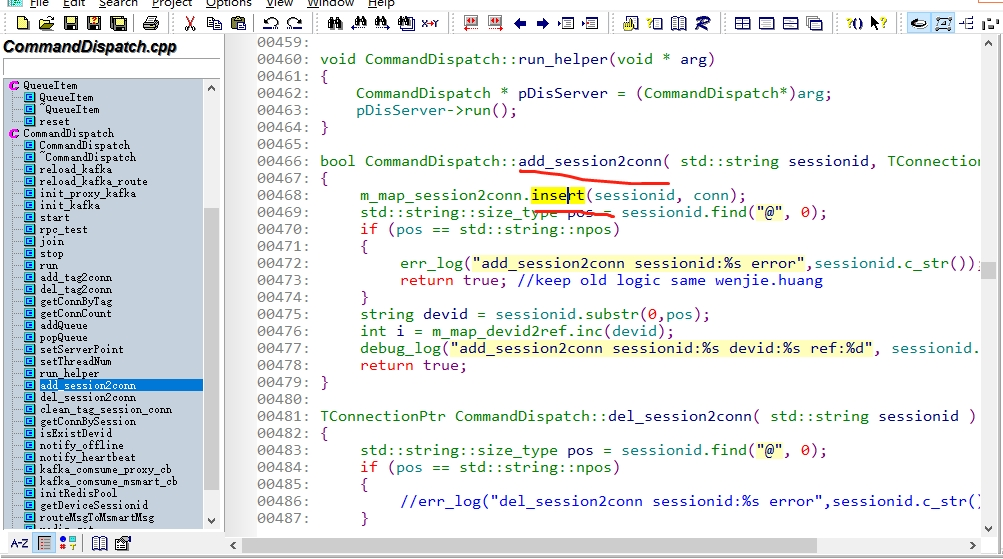
1. 函数运行到aesEncode加密函数或aesDecode解密函数中的AES\_cbc\_encrypt函数时，出现了codedump，

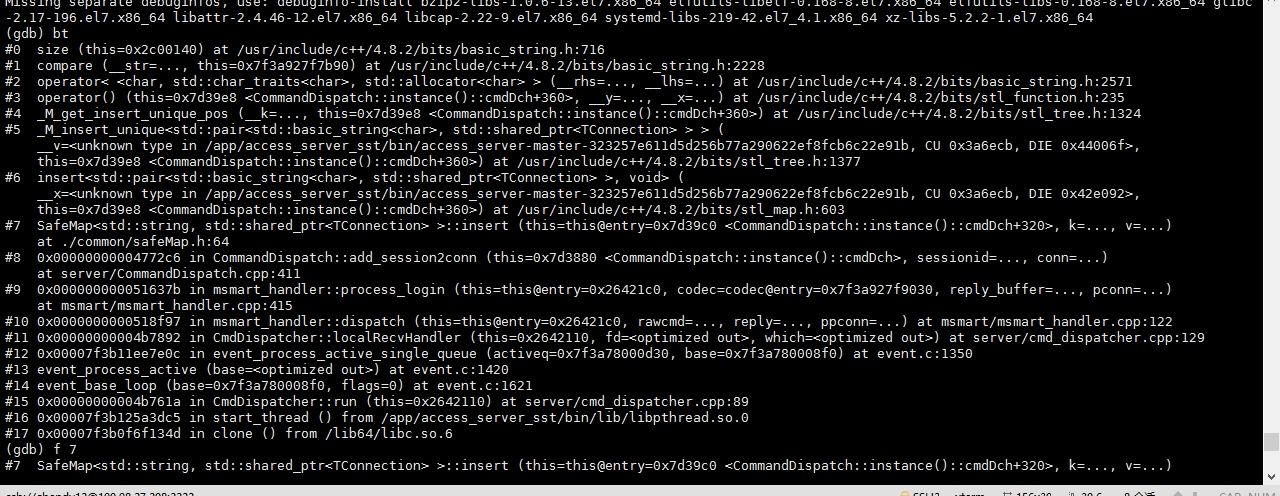


经过查找相关资料然后结合自己分析思考，推测是被加密的长度encryptlen要是phead->Cnt（16字节）的整数倍，否则要手动补齐：**比如块长度为16字节（128位），看被加密的长度是否是16的整数倍，不是的话，用缺少的字节数补齐到16的整数倍(例如：少3个字节，则补3个3)，如果刚好是16的整数倍，则多加16个字节(每个字节均为16)。**

然后经过观察与思考，作出了相应的修改，等待线上的验证。

1. 在把sccession id插入map的时候发生了错误，我提出的建议是再认真检查加锁的步骤，保证每一个可能的步骤都要避免竞态而导致的coredump





## 其他产出

a) 为接入层代码的基本框架编写了codeReview 文档，方便自己以后的查看以及后人的理解

b) 为接入层的大部分主要逻辑代码添加了注释，方便自己以后的查看以及后人对代码的理解

c) 编写了go语言的编译环境配置文档，可以方便部门里想要配置go语言环境的同事。

d) 编写了几种类型的coredump分析过程以及改进过程，方便以后的同学学习

# 个人收获和不足

## 个人收获

**技术上：**

* 熟悉了接入层的代码以及其设计框架，学习了libevent、redis、kafka等框架的使用，对多线程并发服务器的理解加深了，为以后的工作做好了铺垫
* 经过自己亲身的压测体验之后，了解了改变各种参数对服务器的影响，自己对服务器的设计理解也加深了
* 学习了一门新的语言——go语言，自己的技术栈加深了
* 学习了如何用gdb结合core文件快速定位到出现coredump的地方，以便更好地解决问题
* 经过协助导师解决线上的coredump问题，自己对线上问题的发生与解决有了初步的理解，大大有助于我以后的工作

**其他：**

* 收获了经验丰富并且耐心的导师指导
* 收获了良好的同事关系
* 收获了美的集团对“美少年”的关怀——美少年运动会、下午茶、团建活动、各种指导课程（如何适应、提升、提高效率等）

## 不足

* 熟悉业务、学习需要的技术时还不够快、不够全面
* 还未能独立承担起接入层的一些工作

# 发现的问题以及建议

## 发现的问题以及建议

1. 接入层代码的代码量很大，但是注释和文档却寥寥无几，我觉得重要代码的部分需要加注释；同时也要注意文档的记录，因为感觉我们部门也是发展中嘛，就我入职以来就看到了好几个新同事的加入，所以的话有文档的迭代还是比较重要的，可以让我们的工作更方便高效，尽量避免“口口相传”。
2. 刚才讲到我们部门的业务当然是蒸蒸日上的，所以培养新人的话还是挺重要的，所以呢，我想提一下关于实习生培养或者应届生培养的一个建议。就是制定一个培养方案的话对实习生的培养是比较好的，比如安排哪一段时间学习哪部分内容，学习完这部分内容之后要做什么产出来检验，然后学习的内容是贴近以后需要做的工作的，学习完几个模块做了几个练习之后就能很好地参与到工作中。这样子能够更加平滑地学习，效率也更高，实习生的实习体验也会更好。

## 对岗位工作做的改进

1. go语言编写客户端，对服务器的测试服务上得到了优化
2. 协助导师修改线上服务器的错误，使服务器的容错性和稳定性作出了改进
3. 编写了多个文档，包括接入层代码的codeReview文档、go语言的编译环境文档、接入层编译环境文档更新、gdb+core文件的coredump学习笔记，方便了同事或新入职同事的查看与学习，能更好地迭代。
4. 对服务器进行了多组数据的压测，优化了服务器可接收的连接数、连接数的接收速度

# 答辩问题准备

1. **秋招会不会考虑其他公司？**

秋招的话暂时还没有考虑这么多，就现在来说的话我感觉在这里还是挺舒服的，跟同事的相处也挺好的，去其他公司的话不一定会有美的好，所以现在有机会留在这里的话我还是比较倾向于留下来。

1. **为啥选择c++，c++的特点，优势劣势有哪些?**

因为我大学期间开发项目主要用的是C/C++，首先是对这门语言比较熟悉吧，然后用着用着发现挺有趣的，能做的事情也很多，所以就想从事C++这方面的工作。

C++的话最大的优点就是相比于其他高级语言，更接近底层，因此就效率更高，粒度更细，适用于一些要求高性能的服务中，能做的东西也很多。

缺点的话大概就是开发速度可能会比较慢，因为可供选择的库比较少，很多东西都要自己亲力亲为自己写。

1. **有考虑过薪资吗？你期待的薪资是多少？**

薪资的话结合目前应届生程序员的行情月薪一万以上的话我是可以接受的，另外的话能给到多少我相信公司会根据我的能力与付出给我一个相匹配的回报。

1. 在美的感觉如何，愿意留下来吗？

这两个月相处下来感觉还是挺好的，导师、同事都挺照顾我，带我去附近的各个地方吃东西；然后公司的加班也不算严重，就是相对于那些一线的互联网公司来说的话，还是不算辛苦的；然后工作内容的话也是自己比较感兴趣的一个方面，所以总的来说自己还是比较喜欢这里的。