本文涉及到的代码以及框架均来自本人多年的经验积累，设计到的一些框架代码在项目lib文件夹jar中

[無心道（失足程序员） - 博客园 (cnblogs.com)](https://www.cnblogs.com/shizuchengxuyuan/)

[游戏框架设计方案: 游戏服务器框架设计方案 (gitee.com)](https://gitee.com/wuxindao/game-framework-design-scheme)

# 服务器设计结构

# 线程结构

1. **网络线程**

我相信大部分项目现在使用的网络框架都是netty，netty分为boss线程和work线程，

Boss线程是用于socket链接监控，新接入，收消息等状态机处理，

Work线程是用于消息接受转化，已经处理工作，有部分设计中把业务逻辑直接放到work线程中处理，我不是很建议这样做，因为如果你处理的业务耗时又或者在rpc远程调用过程中，当前work线程会卡主并且被netty自动绑定到当前work线程的所有session也会卡主；导致无法接受消息。所以尽量选择业务逻辑的处理跑出去，搞一个专用的业务线程池；

1. 逻辑线程

逻辑线程池，其实就是用来处理业务；逻辑，比如玩家心跳检测，地图心跳检测，服务器活动触发和关闭等业务逻辑，玩家的消息请求也会被派发到此线程池中（注：可以在线程池中加入消息队列）

1. 数据落地线程

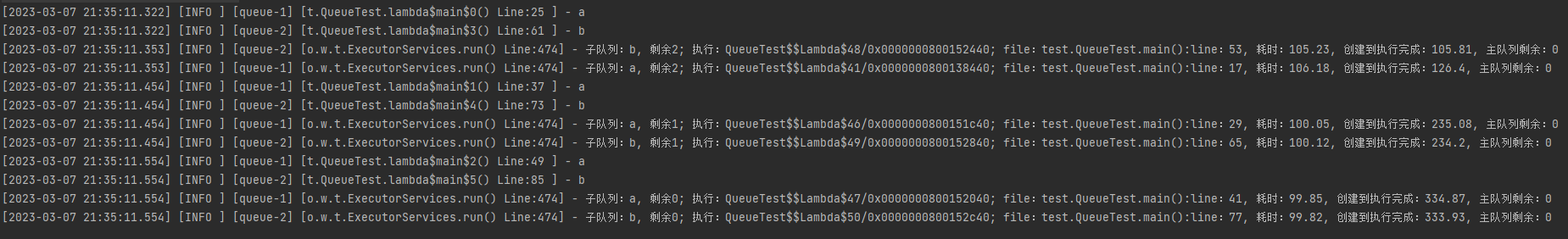
数据落地线程池是为了异步去和数据库做同步，把全服数据，玩家数据，异步写入数据库，因为通常这步可能会比较耗时，所以需要把数据丢入线程池，等待异步落地这样就可以减少业务线程池的消耗带来的卡顿和延迟

1. 消息队列

消息队列是为了解决并发带来的锁消耗，我们可以通过消息队列来避免因为并发需要对公共部分加锁带来的消耗而且还能避免死锁导致整个系统卡死；

PS：线程池的使用或者线程使用应该尽量考虑线程复用，减少线程切换，那么我认为最好的方案是什么呢？就是采用线程池加队列来控制消息流转，线程没有专用线程，（数据落地除外）传统的项目或者很项目的结构中习惯了线程的专用，比如公会, 登录，组队，等都会new一个独立的线程来处理事情，这样就会导致线程忙的忙死，闲的闲死，还会频繁的切换线程导致不必要的开销；我在2014年开始就自己设计了一个闭环队列，线程池里面去跑一个消息队列，消息队列的元素可能又是一个队列，这样线程不要切换，执行完成一个任务去获取下一个任务，子队列当被某一个线程获取之后就会脱离消息队列，等待线程把任务执行完成，把子队列归还到主队列的时候才会继续执行下一个任务；

具体测试用例在 script项目test目录中(queuetest.java)



# 数据库设计

目前在我的设计范畴或者说我经历过的项目中，数据库设计方案大致分为单服单库结构和大服大数据库结构

单服单库，顾名思义就是1个服，对应一个数据库把所有的数据都丢入到这一个数据库中，好吃就是数据量小，出问题概率小，恢复和备份数据特别快；

坏处就是合并数据，数据迁移，也就是传统合服的时候操作麻烦，还有可能出现id冲突等现象，还有一点就是实现跨服比较麻烦，通常会在其中加入缓存数据库比如redis作为热数据中间件

全服数据

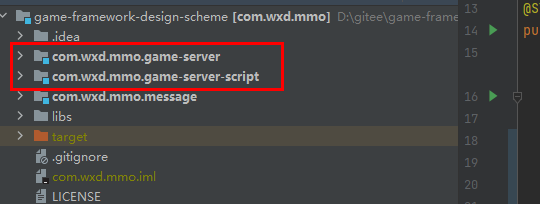
全服数据通常可以定义不同的类型，进行数据类型归类存储；例如排行榜，活动开启，当前服务器注册，充值等详情记录（设计表名的时候用server\_功能名）

玩家个人数据

玩家个人数据建议才有模块进行独立模块划分，独立存储，这样的好处是我需要玩家的数据的时候可以直接加载，比如扣除玩家道具，给玩家发邮件，或者因为活动，排行榜需要玩家部分数据的时候直接加载指定模块就行，不至于加载玩家所有数据进行操作减少数据库io和内存消耗

# 项目的模块设计

在我的设计习惯中独立项目块儿比如，登录，网关，逻辑，社交这样的项目结构，那么



在设计中一定是一个启动项目，加一个脚本项目结构，

比如xxx-server和xxx-server-script

这样的设计的目的是为了用server项目进行进程启动程序用script项目进行业务逻辑实现，这样编译业务逻辑问题的热更，做过游戏的都知道，游戏项目最大一个需求就是绝大部分bug能不停服更新

在此结构下，就需要xxx-server项目下去做数据的缓存。比如xxx-map 然后具体的方法放到script里面

整个项目结构管理，大家可能都熟悉spring，用springioc容器对实例进行容器化管理，自动注入等，也有google的guice框架进行容器化管理，也可自己实现更为符合自己需求扩展性和管控性更高的方案实现。但其实都差不多，选择自己熟悉和常用的方式即可

# 登录模块

数据加载

保证账号唯一性

不同平台Sdk处理

如果有网关负载均衡处理

# 充值模块

平台充值回调处理

推送逻辑服务器订单发货

# 逻辑模块

活动管理

GM管理

状态机管理