《独立竞技游戏前后端框架引导》

李骏

现在竞技小游戏很多，很多人都想要做自己的一套东西，我之前在ULUA王者荣耀开黑群给空白等人说过一些琐碎的指导意见。因为其他人也需要，因此整理了下。

游戏基本分：Unity客户端+服务器端+存储方式，各自简单论述下吧。

# 客户端

其实现在的Unity的开发引擎已经非常成熟，大作不再讨论，我们主要说下独立竞技游戏如果要做需要研究哪些框架。

<1> UI框架，其实现在很多了，NGUI\UGUI\FGUI等。

<3>行为树，目前有很多有腾讯免费的behaviac，自带一套独立的编辑器，王者荣耀都基于它，下载地址：<https://github.com/Tencent/behaviac>

还有个老牌的行为树插件，编辑器用Unity完成的BehaviorDesigner，它收费的，但是有很多免费分享的。

<2>热更新框架，这个也太多了，当然在我眼中我家的最好ulua、tolua，参考框架的话可以

LuaFramework\_NGUI、LuaFramework\_UGUI等。里面都有详细的介绍Lua代码怎么操控c#代码及其Unity的网络交互实例。

<3>我们主要讨论下网络层框架，我熟悉的有两套比较好：

(1)Supersocket 1.6.2(开源) 下载地址：<http://www.supersocket.net/>

(2)Forge Networking(开源)

下载地址：<https://github.com/BeardedManStudios/ForgeNetworkingRemastered>

这里论述下他们的有利弊，首先他们两者都支持TCP+UDP模式。

**集成性比较：**Supersocket是独立项目跟Unity的集成不是很紧密，需要自己做集成，还好它本身提供了c# client代码，或者我们自己用c# socket独立完成都可以。

而Forge Networking本身依赖于Unity，集成性非常紧密，所有客户端实力代码都很丰富，这一点Supersocket不如Forget Networking。

**完善性比较：**Supersocket在国内已经有众多游戏厂商应用到上线游戏，不管TCP模式还是UDP已经非常完善。而且经过验证，目前的1.62已经是非常稳定，值得推荐大家，不管是断包、分包都可以根据它定的规则非常迅速完成。

而Forget Networking目前还没有明确使用它的项目上线，而且它的TCP模式作者并没有很完善，具体表现在，如果服务器发过来一个很大的Package，客户端缓冲池并没有办法一次性接收，等下次接收包头实际并非真正包头，而是断包的后半部分，所以会直接崩溃，我试着想自己修复，可惜它代码流程结构紧密，有点难以下手，后来只能换做UDP模式因此这是个很大的缺憾。但是王者荣耀等众多竞技游戏基本都会选用UDP模式，因此这块Forget Networking不如Supersocket。

**跨平台性比较：**Supersocket跟Forget Networking两者都是基于c#代码，因此理论上没有过多差异，唯一区别，Supersocket的服务器文档比较完善，可以直接基于mono运行在Linux、或者Windows Server。其实Forge Networking也一样，他们各自提供了一个Server引导的壳，也是可以跨平台，但是这块文档欠缺。除了mono，微软已经推出.Net Core平台，因此我们可以直接使用它，毕竟mono是个山寨品，而微软才是原版的，完善度、运行效率还是微软更胜一筹。

Mono地址：<http://www.mono-project.com/>

**.**Net Core地址：<https://www.microsoft.com/net/core#windowscmd>

<4>素材商店：首选大淘宝，你可以在上面搜索Unity素材，各种模型、动画、粒子特效、原图很多。还有就是Unity AssetStore地址：<https://www.assetstore.unity3d.com/en/> 这里可以购买很多用户原创的素材，模型、特效、功能性插件，并且开通了支付宝支付，很方便。

# 服务器端

服务器这块基本逻辑就别搞那么复杂，初期研发的时候，基本一个Matster+Game Server即可，如果细分的话，服务器会有N多分支，一时半会儿也讲不完，我们就讨论个最简单的。

Game Server

Game Server

Master Server

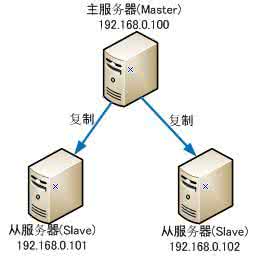
MasterSever一般可以作为服务器管理器服务器，接入服务器，登录服务器等，验证服务器，等非逻辑性服务器。而GameServer可以在上面根据硬件资源开辟房间，比如王者荣耀的一局排位、匹配，就是把所有角色直接塞到这个房间里面，从MasterServer做过安全验证后的角色，可以保持连接，进入新的GameServer，或者断掉MasterServer，再连接新的GameServer，这样都可以，根据需要来做。

服务器框架：一个是我提供的ServerFramework（基于Supersocket），另一个还是Forge Networking的独立服务器框架，包括一些移动状态同步插值运算都内置了。我们需要完成的就是一些接入逻辑，分发连接，游戏逻辑等。

# 存储端

**持久化存储：**服务器数据都需要存储，目前国内比较成熟的一般都是MySQL+redis模式，MySQL就是非常老牌关系型数据库了，免费+高性能，（MSSQL就不讨论了）<https://www.mysql.com/>

简化的一个MySQL实例即可，稍微扩展点的就是MS架构（Master Slave）



Master主服务器主要接收“写操作”，另外的Slave服务器主要提供“读操作”，这样“读写分离”性能会提高很多，相当于一个双向马路多么的拥挤，但是两条并行单行道，就可以很大分担读写压力，而且可以起到备份效果。就简单说下而已，再复杂了就是水平分割、纵向分割数据库集群，这里就不讨论了。

**NOSQL高速缓存：**MySQL虽然性能很好，但是由于众多原因还是不能满足大并发流量的操作及其高速读写，因此我们需要一个内存缓存层，你可以选择完全基于内存的Memcached，下载地址：<http://memcached.org/> 也可以选择CouchBase、Mongodb等。类产品上千种，数不胜数。

目前国内应用非常广泛的当属redis，下载地址：<http://www.redis.io/>、新浪、百度、腾讯等大公司都有众多的技术积累，现在游戏公司开发也基本从MySQL+memcached转移到了MySQL+redis。

Redis具备超高速读写缓存，你可以通过KeyValue方式快速读取，而且支持的数据类型比mc（memcached）要丰富很多，2.8版本还支持内部lua查询语言，你的query的结果集不需要拿到GameServer在redis进程内部完成查询，将结果返回。3.0的版本后开始支持MS架构（Master Slave）集群。你可以布局成高速缓存集群，对服务器的容灾做好准备。

当然除了高速缓存，redis还支持持久化，你可以将数据落地，它一般支持两种，一个是rdb全镜像备份，另一个aof的定时策略备份，但是恢复的时候rdb速度要比aof快，目前没看到改善，所以一般公司都拿redis做高速缓存，而不用它的持久化。