**调优计划：**

1. **本地加速（不采取异步的方式）：**

**lua层优化：**

**a)字符串操作转移（譬如不依赖于同步逻辑的提示类均可发到其他service去完成拼接和替换然后再发给客户端，暂时可放在broadcast临时实现该转移）**

**b)发包encode/decode转移（这个不需特别调整，依赖于第3点完成即可），一方面提速，一方面减少重复register/unregister各个service传入mailladdr的过程**

**c)遍历类发包利用broadcast（尤其是公会/好友这种存在较大遍历量的，仔细检查）**

**d)baseobj对象是否采用缓存池，甚至于针对大量分配释放的对象（譬如道具）形成定制的缓存池，减少分配和释放耗时，另外降低GC压力（该优化需谨慎考虑，空间换效率，另外需特别小心复用对象导致的副作用）**

**e)主城过一遍自己项目netcmd的协议分布，分析哪些地方值得优化，譬如H7之前的背包整理功能优化过后下降了4-5倍耗时**

**f)注重lua编程中的高性能技巧，这里需要整理一份文档**

**C层优化：**

**一般业务还是使用lua，但涉及大量计算，譬如背包整理、属性计算可考虑抽象出简单的计算参数传入C层进行计算**

**编译器相关优化：**

**由于目前项目采用lua5.3，无法得到luajit加持，可以考虑小面积改动lua编译器：**

**a)lua-checksig是否去除**

**b)lua编译选项问题**

**c)表分配初始化大小问题，是否设置一个较大值，空间换效率，降低realloc概率**

1. **远程加速（采取异步的方式）**

**独立逻辑体的抽离：**

**譬如scene、war等都属于该类，目前还存在未抽离的公会也是属于此类，此类逻辑运行一般大量依赖于自身，较小依赖于其他逻辑体，抽取出去难度小、性价较高**

**协助逻辑体的抽离：**

**这一块我们在积极做尝试，基本理清流程，考虑到可能的坑。大概的思路：以玩家为例子，将玩家部分逻辑较高内聚的属性块（较少与世界其他物件交互），譬如道具、宠物、伙伴、技能、坐骑等等，都可以认为是此类，其中按目前MMO业务模型，道具系统的转移是具有较大难度，其他抽取难度尚可，性价比较高。此类逻辑体，需谨记自身只是作为主逻辑体的协助者，一般不需拥有完整功能，帮助主逻辑干完事情进行response或者主动notify即可，若需主逻辑之上的数据则一般由主逻辑调用之时捎带过去即可，而主逻辑可通过共享内存访问到协助逻辑体之上的部分简单数据结构。**

1. **网络包的收发整理**

**网络收包的合并和管控：**

**网络收包进行定期收集，定期提交到login/world，减少多次处理消息的头部消耗.另外，针对玩家相关的连接进行包频管控**

**网络发包的合并和集中：**

**减少流量消耗，优化客户端收包效率，另外方便各个上层service针对pid进行发包，不需要关注mailaddr的变动**

1. **硬件加持**

**按目前UC物理云主机和云主机的对比，云主机的计算效率和稳定性方面都较差（存在与其他云主机的资源竞争，参考公司其他项目因为该问题，内存oom事件已经有八起），分析二者的性价比，可以考虑购置一个物理云主机，然后同主机上部署多个GS或者CS实例，从成本对比而言只是稍有上升，却大大提高稳定性，且减少项目成员的人工成本，值得商讨**

1. **采取分布式结构的其他尝试**

**以玩家为主体进行逻辑切分，将玩家所有数据都统一抽离，并将玩家所在service作为关键状态发生的第一现场，而主world上面只维护玩家的部分必要简单数据.这样做的好处在于玩家service可负载均衡去消化网络包，并把部分只涉及玩家自身的包处理截断在自身所在service，不向后传递到主world;坏处在于分布式程度比目前更高，所需写的异步逻辑更多，暂不深入探究，可在新项目讨论可能性**

**目前状况：**

**按目前在H7测试情况，抽查部分常用客户端发包处理，譬如组队/获取道具等，常规操作大概是0.1-0.5ms级别，有部分逻辑较低或者纯转发接口譬如同步位置大概是0.01-0.05ms级别，而部分较为耗时的譬如整理背包达到了1-10ms级别.我们模拟走路(3K)+模拟战斗(1K)+模拟常规操作(4K)的脚本测试，大概外服的承载力大概5000-8000左右，这个测试不排除脚本方面会有些严苛（譬如一直在得到道具一直在使用道具），所以常规操作脚本对我们的考验较大，但总体上提醒要优化主world的计算效率，使用频率较高的功能要么本地优化加速（优化算法、C层加速），要么通过功能切分到其他service.**