任务服务器设想:

目前,任务比较复杂,配置比较容易出错,现在我设想了一种任务服务器,结构如下:

假设:

1,玩家不需要别人的任务数据,只需要自己的数据

一,任务本身:

任务服务器基本是一个被动的服务器,只有3个直接操作(可由玩家发起的操作):

创建(接受)任务

撤销任务

查询自己的任务列表

更新任务数据的工作,主要由服务器内部的通信来完成

任务数据:

任务数据为protobuf/json格式的字符串,即,任务服务器自己不负责解释该数据,这些数据交由lua脚本解释

更新数据的工作,其实就是Lua调用服务器接口,更新任务数据的字符串值.

由此,可知任务数据库表的内容比较简单:

taskid:任务id

taskdata:用户数据.text格式(服务器可能需要估计下这个格式的大小,即,策划的功能所需数据,会不会超过服务器允许的数据大小?)

更新数据的操作:

calllua("functionName", taskdata, ...);//提供函数名称,taskdata和其它参数,

比如,扣除道具的函数,可以写为

calllua("itemUpdated", taskdata, itemID, itemNum);

注意,服务器需要提供lua写taskdata到数据库的支持.(提供一个函数调用就可以了)

lua编写的时候,需要注意不要使用任何全局和局部变量,所有数据全部保存在taskdata里面!这一点需要策划注意,(这么做的目标,是保证lua可以重复加载,从而实现热更新)

举例:

玩家消费3个道具,就可以领取奖励(奖励需要手动点击领取)

function itemUpdated(taskdata, itemID, itemNum)

local taskItemUpdated = false

if not taskdata.item1complete and itemID == itemid1 then

taskdata.item1complete = true

taskItemUpdated = true

elseif not taskdata.item2complete and itemID == itemid2 then

taskdata.item2complete = true

taskItemUpdated = true

elseif not taskdata.item3complete and itemID == itemid3 then

taskdata.item3complete = true

taskItemUpdated = true

end

if not taskdata.waitforaward and taskdata.item3complete and taskdata.item3complete and taskdata.item3complete then

taskdata.waitforaward = true

taskItemUpdated = true

end

updateDataBase(taskdata) --这个函数到时候可以做成自动更新的,最后一句update就可以不需要了

return taskItemUpdated--通知服务器任务数据有更新

end

这样,服务器运行时,玩家身上可能有多个任务.收到其它服务器发过来的itemUpdated消息,直接对所有的任务执行itemUpdated函数.

查询任务列表:

服务器使用taskdata参数调用lua函数,lua函数返回客户端可识别的数据,服务器转发到客户端

玩家任务奖励:

调用lua函数即可,服务器需要提供常用的奖励lua函数(加经验,加道具等)

注意1,由于服务器不能解析taskdata,所以lua函数需要自己判断任务是否完成.上面的示例代码,就是使用了一个独立的数据库字段waitforaward来表示任务已经完成的.

注意2,有些任务是自动奖励的,这些就不需要注意1里面的独立变量了

任务更新noti:

以上即任务的设计.

二,外围设计;

1,更新:

假设服务器为单线程程序:

使用gm指令,重新加载所有lua文件即可.按照上面的Lua文件的写法,是没有任何数据的,可以直接删除lua虚拟机再new一个

1.1,不能重启任务服务器,因为可能服务器还处于忙状态(比如某个脚本正在给玩家发奖励,一重启,奖励就可能发送不成功),gm指令的好处就是保证服务器是在空闲状态才更新lua文件的.

1.2,很明显任务服务器可以做成多进程的(因为没有数据库以外的其它数据),因此暂时不需要考虑性能问题,如果有性能问题,就按照多进程来跑

2,某些限时任务:

调用Lua脚本的时候,设置时间参数

3,自动接取的任务,比如某些活动:

待定.基本思想是根据配置的时间,提前给所有玩家创建该任务

三,protobuf的一些注意事项:

因为lua可以自己解析Protobuf文件(已知taskid,因为Lua脚本内部可以写死自己的taskid),所以,如果使用protobuf作为taskdata的格式,热更新仍然有效,只是服务器日志可能不容易看出来.(taskdata在服务器日志中会表现为乱码)

四,Lua脚本:

1,不能使用任何变量,包括全局变量和局部变量!所有数据都保存到taskdata!!

2,脚本回调函数:(服务器调用lua)

function init(taskdata) --初始化taskdata的函数,理论上用不到(数据库应该可以自动初始化会为空串),看情况可以从文档删除(实现目标:一定要删除)

function xxxxUpdated(taskdata, updateditem, ...)--各种事件响应函数,更新taskdata用

function award(taskdata, ...)--奖励函数,因为有时候,需要允许玩家手动点击获得奖励.如果用到这个函数,需要检查玩家是不是真的有权领取奖励

function getTaskDesc(taskdata) --获得任务的描述,可以发给客户端,或者服务器记录日志使用

3,服务器支持函数:(lua调用服务器)

void saveUpdate(taskdata&);

void addXXXAward(int coin);

4,其它:

Lua脚本可以检测taskdata是不是有更新,从而在没有更新的时候不通知服务器

刚开始可以由程序员来写Lua,写好之后,策划复制Lua脚本,修改参数即可