

搜狐公众平台 >IT

他一出家就成中国最帅和尚
眼眸深邃、轮廓分明、身材颀长，活生生的一幅画。

大学副教授与在押服刑女结婚
这在监狱民警看来，那么令人不可思议。

原创 《梦幻西游》手游主程刘强：如何提供一个稳定高效的MMO手游服务器引擎

手游那点事 2015-06-30 10:49:37 阅读(0) 评论(0)

声明：本文由入驻搜狐公众平台的作者撰写，除搜狐官方账号外，观点仅代表作者本人，不代表搜狐立场。

举报



整理/手游那点事子安

随着手游越发端游化和重度化，衡量一款手游的数据除了传统的留存率、ARPU等外，还新增了PCU（Peak concurrent users，既最高同时在线玩家人数）等数据。但是同时多人在线对服务器的运算能力和稳定性都有着极大的要求。在此次的网易游戏学院上，《梦幻西游》手游的主程刘强就《梦幻西游》手游中如何做到200万人同时在线，如何提供一个稳定高效的 MMO手游服务器引擎等问题发表了演讲。

以下是演讲整理稿：

一、《梦幻西游》手游引擎水准：市场唯一过万单服架构

刘强：大家好，我是刘强，也可以叫我“强宝”。有人问我说，为什么大家讨论都客户端技术，移动端的技术没有人讨论。好像是这么回事，第一眼美女，大家比较关注，但服务端同样重要。所以我给大家介绍一下《梦幻西游》手游架构是什么，以及为了保证架构比较稳定，我们又做了哪些努力。

服务器架构，我们手游引擎大概是怎样的水准？我们的引擎来于梦幻端游，大概2.2万单服架构，手游引擎稍微差一点，1.5万。但是我们还是有比较大的优化空间。据我了解，在市面上还没有过万的手游引擎，在手游单服架构里，当然除了我们。



手游那点事
手游那点事，关注移动游戏运营和推广，每日发布手游行业重要新...

| | | |
|----------|-------------|----------|
| 0 阅读量 | 2790 文章数 | 0 评论数 |
|----------|-------------|----------|

原创 《梦幻西游》手游 搜狗搜索

热词：死亡航班 饲养蜘蛛侠 夺命房间 引力双眼皮

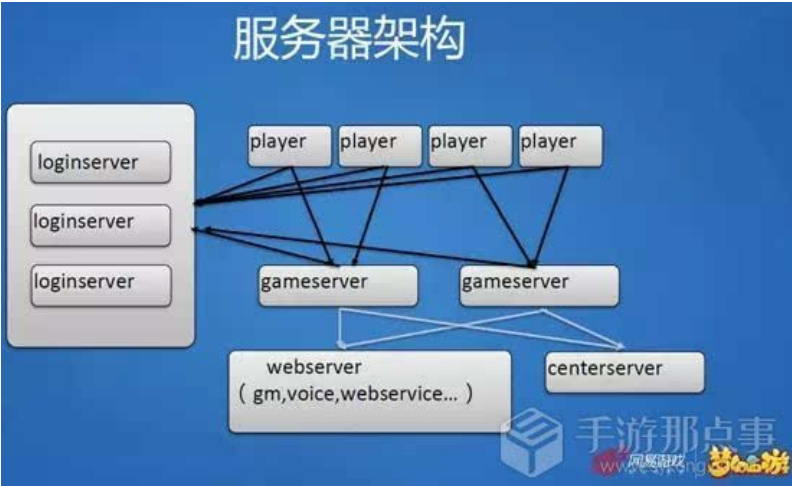


如何省去装修中最大的那笔开销？

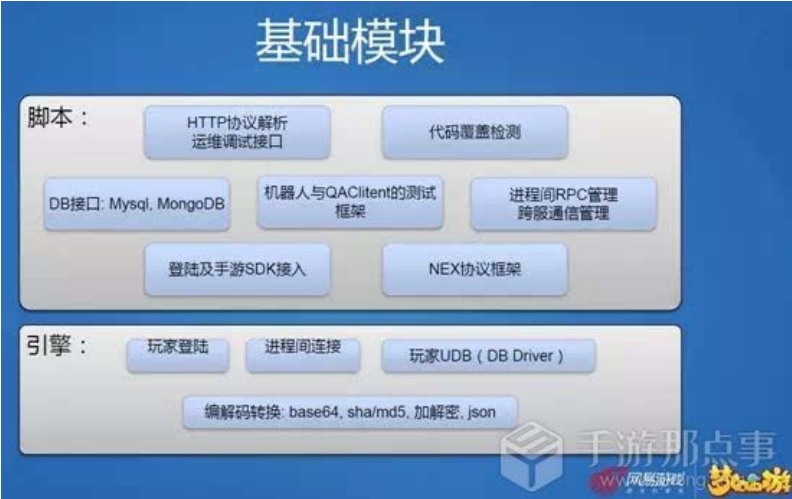
热点视频 影视剧 综艺 原创

我来说两句排行榜

客服热线：86-10-58511234
客服邮箱：kf@vip.sohu.com



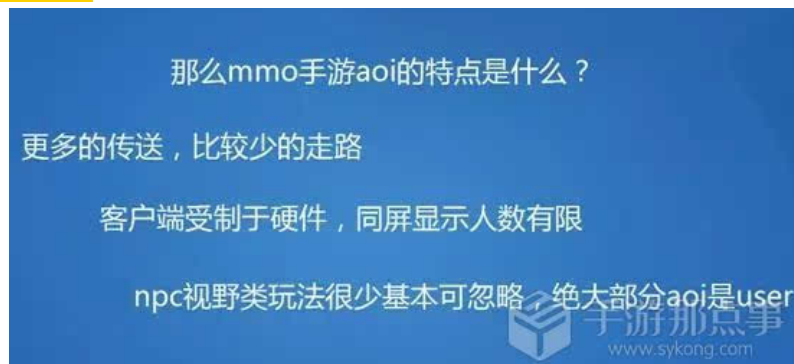
这是我们服务器架构，主要分几块，有gameserver、loginserver、webservice、centerserver。首先是loginserver，loginserver访问了以后，进行一些帐号的验证，然后根据验证后获得了列表后，然后开始访问Gameserver，Gameserver自己跟所验证的server进行验证，这样就保证了我们loginserver可以无限增加的。当时考虑loginserver可能会负载高，所以设计成这种架构。然后Webserver顾名思义，就是一些gm、voice用的，因为我们提供语音聊天和语音转文字服务，这应该是社交类游戏的神器，一定要提供语音服务。Centerserver就是全服玩法的活动。



引擎线程结构，我们引擎是多线程的结构，分线程的理由，首先模块比较独立，然后很好分。虽然我们多线程引擎，但是开发时思路用单线程来开发，所以并没有损失我们的开发效率。我们大部分是一些IO和CPU密集型的线程脱离，主线程主要是游戏逻辑，这也是一些基础模块，就是我们分为引擎和脚本，脚本里主要跑逻辑，引擎里主要是跑基础的像玩家登陆，还有就是CPU密集型的东西，放在脚本里比较合适。

二、MMO手游AOI特点：更多的传送，NPC视野类的玩法少

手游和端游AOI有明显的区别，有两种算法，一种九宫格算法，插入和删除节点，速度非常快，属于O(1)。但是有一个很大的问题，就是当它在格子间移动时有大量的带宽消耗，因为有很多无用的信息在发送。十字链表，移动时是很省带宽的，但是在长距离时，它的CPU是消耗非常大的，这两种我们网易这边都有实现，都尝试过，也对它们比较了解。



MMO的手游AOI有什么特点？更多的传送，都是属于直接传送到地图中间，比较少的走路。客户端受制于硬件，同屏显示人数有限，不像端游，可以随便显示很多，这样需要显示人数减少，否则手机会卡死。NPC视野类的玩法很少，基本可以忽略。就是说十字链表在做NPC视野的玩法是非常好的，它所有的NPC可以选择不同视野范围，但是我们手游基本上这种玩法不考虑，因为没有嘛，所以我们选用格子。

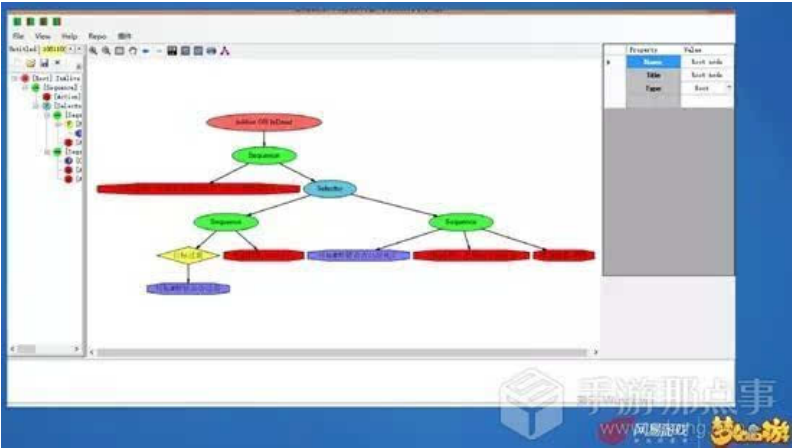
这是我们开发到中期时，然后在做1万人压测时的效果，这些全是机器人，他们在里面还是走路，这是我们做不删档测试第一天的时候，因为没有放AOI优化的效果图，这是真实的玩家，就是相当于万人以上的，大概效果是这样的，和压测的效果是一样的，都是满满的人，所以整个世界上都充斥着各种“吐槽”的话，所以玩家体会很糟糕，因为客户端卡，手机上显示这么多角色是非常糟糕的一种表现。

鉴于这种，我们做了一个分层AOI概念，不同的玩家，还有不同的组队信息，以及好友有一个很复杂的规则，把玩家分在不同的层次，还有就是当人数增多，可以动态进行分层，人数少的时候可以把层数合并回来，然后就是让玩家在人少的时候也能看到几个人，人多的时候还是看到几个人，所以不至于很少。这种优化放过来后，这是同一台服务器，效果大概是这样的，你在同一层里看到的都是你关心的人。这个感受就很好了。

三、基础模块的实现：udb以及AI行为树

为什么要设计一个udb，udb有做统一的管理，在定时存盘+退出存盘，经典解决存盘的模式，有一种性能瓶颈，当你的同服玩家最高在线没有很高时，是不会遇到这个瓶颈的。在定时存盘时，同时想让一万以上玩家做存盘，这时候序列化会很卡，很耗CPU，并且有down机回档的问题，有一些人已经退出存盘，但是另一些人在等待定时存盘，当Down机，两个人的DB是不一致的，所以这很多运营上的问题。运营又给我们提了一个需求，玩家数据备份能不能随时随刻到任意一点，因为我要拿到那一点玩家的数据进行处理。我们现在UDB可以支持这个了。还有一个硬件条件，现在64位机器内存足够大，缓存一周所有登陆的玩家。

AI行为树，因为我们是回合制战斗，所以AI行为树非常适合我们。我们为什么要选择AI行为树。因为非数值策划也可以很容易学会写AI，说起来很容易，如果不使用行为树是很难的，我们现在连文案策划也可以写一些简单的AI。当时项目情况，在有一个端游或者有一个竞品参考时，策划工作稍微小一些，但是程序开发量很大，所以把所有能分给策划写表或者填图的都分给策划做，这样分工合作才能整个提升团队的工作效率。



这是我们的一个AI行为树的工具图，有四到五种节点，就很直观的先干什么，再干什么，添了这个后，所有的AI全部搞定了，所以程序基本上不关心战斗了。

后来我们做一些统计，现在我们基本上所有战斗采用行为树，在战斗力消耗的CPU里，大概超过50%消耗在行为树上，目前我们现在战斗没有遇到瓶颈。

四、优化引擎从CPU、内存、IO以及延时入手

我们的针对手游改造后，怎么提升引擎健壮性，主要是从几方面考虑，首先内存，你的引擎稳定，内存越界写，内存泄漏是必须要清掉的。valgring神器，脚本虚拟机需要有统计object的数量，每个object的大小，array、toble等都有总的大小统计。

cpu

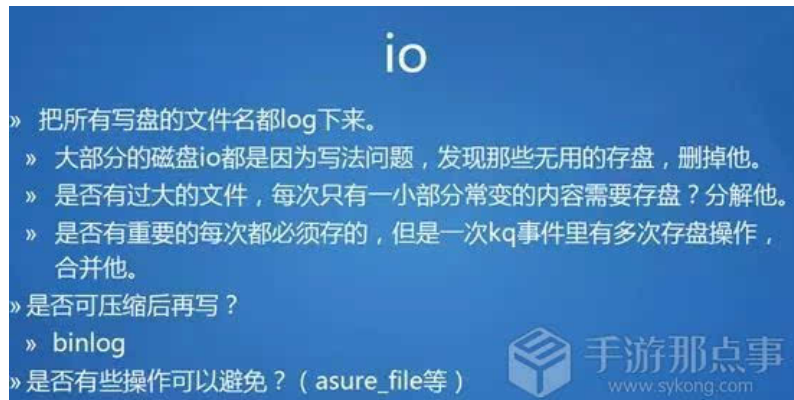
» 首先在上线前你要预估自己的热点可能在哪里？

» 我们分别压测过5000人登陆，5000人走路，5000人战斗。 - 实际的效果会比压测好一些。

» 分别做引擎和脚本的profile，找出你的引擎和脚本热点。热点简化，脚本热点可以引擎化。

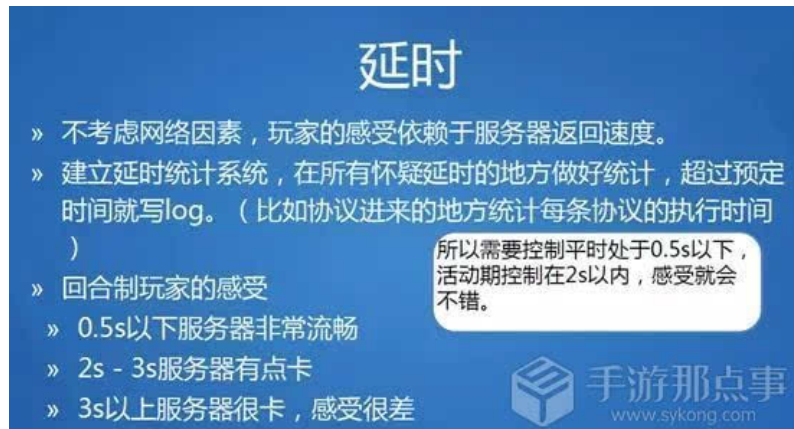
» 脚本做profile的不多，可以考虑在虚拟机调用函数的时候来做profile统计。

cpu，解决了IO就CPU，基本上是一个循环优化的过程，优化了CPU，下一个发现卡的时候就是IO，再优化完IO，又卡到CPU了，是一个循环过程。我们做的这方面的工作，上线前要预估自己的热点是哪里，我们就很好预估，首先登陆，走路、战斗，这是预估可能出现的热点，我们就做压测，先把5000人一压，拼命的干同一件事。压测的效果乘1.5到2倍，假设5000人压了，基本上1万人也没有大问题。另外就是分别做引擎和脚本的profile，找出引擎和脚本的热点，热点简化，脚本热点可以引擎化。脚本做profile的不多，可以考虑在虚拟机调用函数的时候来做profile统计。这是很准的，定位到函数领域时，基本上你的热点也找到了，这是属于解决脚本的一个神器。



IO，实际是一个很大的问题，我们效果最好的一个解决方法，就是多线程的异步写，解决顿卡的超神器，基本上同步读了一个IO导致的，所以我们现在正常的IO都属于异步的，LOG属于异步写的，数据文件存在异步写。需要读数据时，该函数挂起，等到子线程读好数据后再回调该函数。

这是我们一个很简单的例子，因为是一个异步，要注意所有的条件检测，为什么要到原函数，因为函数里有一些条件检测，如果不是直接回到原函数，有可能有一些条件检测会被跳过，这时候可能出现逻辑上的问题。还有就是查IO，我们在优化的过程中会发现，基本上引擎本身的问题是很少的，大部分都是大家写代码的时候写得不够规范，或者是有其他的问题。我们现在主要发现的大部分磁盘io问题都是因为写法有问题，写完的文件我们有记录下来写完的时间和它的存盘名，然后在这个统计下，基本上就可以看到，每一秒存盘的文件，它的大小可以看到，会发现那无用的存盘，删掉它。有可能因为写法不太规范，或者写得比较随意，这个操作就很频繁，就会导致IO的问题。你的游戏里的内容在同一次循环里，有多种的存盘，可以把同一个文件合并到一起，储存最后一次。还有一些就是有哪些IO操作不必要，因为我们每一个文件写盘，都会进行一些asure file，这些可以避免掉的，当你做过一次后，后面就不需要再做了，IO主要就是“省和砍”。



延时，所有游戏的玩家的感受，游戏这种交互感受，他的感受主要依赖于服务器返回速度，所以基本上可以建立延时统计系统，比如说玩家协议过来，开始记录时间，然后给他进行了回复，记录一个时间，记录两条协议执行的时间。以回合制游戏为例，0.5S以下服务器延时，玩家感觉是非常流畅的，没有任何卡顿的感觉，如果2-3秒的延时，玩家有很明显的卡顿感。所以控制平时要在0.5S内，活动尽量控制2S内，感受就比较好。前面是介绍一些我们引擎的一些相关东西，后面就介绍为这些引擎提供的流程上的帮忙，协助保持引擎稳定性。

五、流程上提供助力，消除掉低级问题

codereview，对解决低级bug有非常好的效果，一种强限制型的，还有一种弱限制型的codereview，要选择一个适合自己项目的方式。还有平衡工作量带来的影响，我们项目初期绝对不需要Review的，因为大量的代码，还在堆量，是没有时间做Review的，当项目进中后期才可以考虑开始做这个事情。强限制型有打断的问题，我的工作必须另一个人看完，我才能提交，所以这种限制性比较强。弱限制型，我提交完，让Boss看，看完有问题我再改，我们是采用弱限制型，但是这也不一定，就看自己的项目，采取适合自己项目的情况就好了。

代码覆盖率，主要就是保证QC的测试用例写得足够全面，当我写了一篇代码后，让QC测，并不知道我所有的代码所有代码逻辑都跑过了。每一个内容建立一个项目，比如说这个我们现在还没有跑，所有的我修改的函数都是运行的，这个还没有跑到，这里用例缺少了，就要针对这个再写一些用例，这主要是协助QC用的。

静态代码检测，因为有一些东西跑不到，主要是为了查找是否有笔误，调用不存在的函数，或者调用的参数不符合要求。因为有一些参数写错了，编译又不报，但是你放给外服后，他才开始报错。

自动化回归，为了保证前人种树、后人乘凉，这是一个积累的过程，每次放出版本前进行脚本自动回归，保证已有的东西没有问题，新增内容不会影响到已有内容。

协议测试，这是很重要的，所有的协议必须上QC做一遍，他来设计数值，发送给服务端，测试服务端健康性。尤其手游开发，经常一个人服务端、客户端都做，这样就很容易信任自己客户端传来的数据。当你新写一个游戏后，一定有好几个地方是这样，然后一定会被玩家刷爆。我们最后把所有的协议函数，程序员排查了一遍，是不是所有协议参数都进行了合法性测试，所以这个检测是最重要的。

声明: 本文采用CC BY-NC-SA 3.0 协议进行授权

转载请注明来源：手游那点事

本文链接地址：<https://www.sykong.com/2015/06/70342>

阅读(0)

举报


0 喜欢

0 没劲

分享到


相关新闻

热门关注

搜生活

+关注

搜狐公众平台官方账号

MAGIC杨梦晶

+关注

生活时尚&搭配博主 / 生活时尚自媒体 / 时尚...