高负载低延时游戏架构

【写在前面】

1、高负载与低延时通常是一对矛盾，但现在的客户端游戏，像LOL这样的，单组服务器人数在10万左右，而且对战游戏对实时性要求也很高。如何同时兼顾两方面现在已是很普遍的需求。

2、软件需求通常可分为功能需求与非功能需求，软件架构本身往往并不由功能需求决定，而是由非功能需求决定，比如本文所探讨的架构主要关注的是高负载与低延时需求。

3、不考虑技术相关内容，我们可以把软件系统分为3个抽象层：业务无关层、业务抽象层、应用层。

业务无关层的重用是非常好的，这一层我们基本上都是用别人提供的库，这一层通常是工具软件或系统软件公司专注的核心。

应用层的定义就是特定应用相关的，所以通常是不能跨应用重用的，这一层一般来说是项目组关注的重点，特色功能需求通常在这一层实现。

对行业软件，业务抽象层往往是各行业软件公司的核心竞争力之所在，基本上每个公司都会致力于构建自己的可重用的业务软件库。但这指的是传统软件公司，对于游戏业，我现在怀疑是否存在这一层，原因在于对行业软件所处的行业，行业在本质上有很强的内驱力促使行业形成成熟稳定规范的业务，长远看业务就是能抽象出来的，而游戏业与此正好相反，所有游戏设计者的潜在目标都包含设计出与别的游戏不一样的游戏。基于此，我个人认为，游戏软件系统设计时不宜将重用作为主要目标。

4、对于游戏系统，至少有三类人员与游戏系统关系密切，一类是玩家、一类是研发人员、一类是运维人员。对于这三类人群，关注点不太相同，玩家关注的是功能（主要关注点）与性能（影响感受）、研发人员通常更关心是否易于修改与扩充、运维人员则更关注部署是否方便、是否易于维护管理。

【参考系统或架构】

1、LOL

大厅与游戏分离

基于AIR技术构建大厅

基于ejabberd构建聊天

自主研发游戏客户端（当然还有游戏服务器端）

没有用p2p（来自谢鉷的研究）

\*有个问题是大厅是窗口应用，游戏默认是全屏独占应用。启动游戏有点慢。我们考虑采用基于web的大厅方案，可以把游戏客户端嵌到web里面，也许可以改善这个问题。

2、1018/DNF

（没玩过DNF，所有信息全部来自1018项目）

p2p通信

基于时间戳的同步（从1018学到的概念）

属主机制

服务器验证

3、web

REST架构风格（representational state transfer）

\*URL/URI

\*无状态

\*缓存

\*代理

\*分层系统

移动数据而未移动计算

方便有效的服务器负载均衡

超大规模的可伸缩性

异构系统集成

4、云计算

google三大论文

\*Map/Reduce

\*BigTable

\*GFS

移动计算而未移动数据

本地计算

高可用性

容错与高可靠性

5、自动控制系统

自动控制系统通常不是用软件实现，但这里有非常值得借鉴的基本原理---反馈调节，对于稳定系统调节，通常需要的是负反馈。经典控制器:

\*设定点（SP）

\*比例积分微分调节（PID）

\*输入/输出回路

【架构初探】

㈠基本思路

1、利用不同的基础架构分别应对高负载与低延时要求。利用服务端并行与负载均衡实现高负载（REST、移动数据），利用冗余本地逻辑计算实现快速响应（移动计算）

2、每个客户端是一个确定性自治系统，在没有玩家干预时确定的随时间运行（可预测、可重演、可反演），前提是客户端不作弊

3、对于各客户端模拟的误差，由服务器  校准（SP），客户端进行自动控制调节（PID）。

4、玩家存储数值变化以服务器为准，客户端可以有自己的本地暂存并可用于本地计算，但不影响存储，也不可由玩家查看。另外，所有pvp相关数值或可见结果数值一律以服务器计算为准，客户端依赖此数据的行为（通常是结果数值表现）必须滞后生效或回退误差。

㈡方案概要

1、高负载与低延时分离

2、转发与并行模拟

3、服务器分级响应机制、不同级别延时不同

4、分布式独立AI与战斗逻辑

5、pvp实时判定，pve延迟裁决与同步

6、信用评价与惩罚机制

㈢示意图

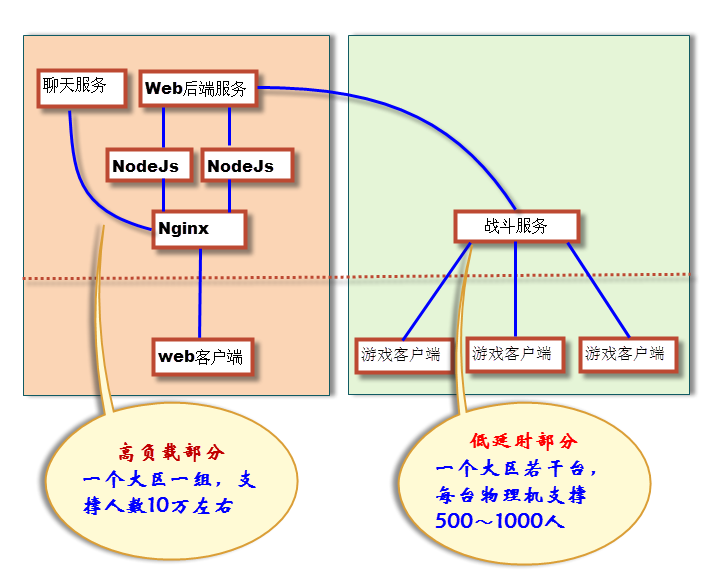


图1 高负载与低延时分离（web+客户端游戏）

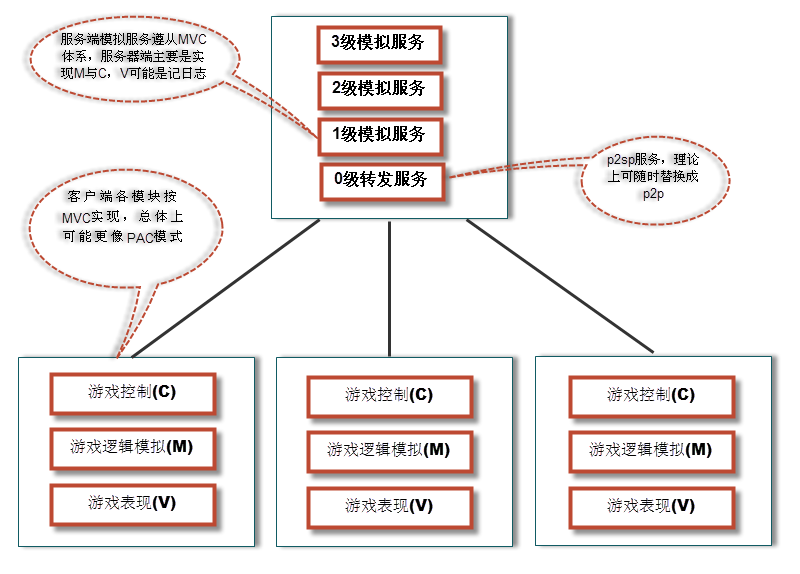


图2 从MVC角度看低延时架构

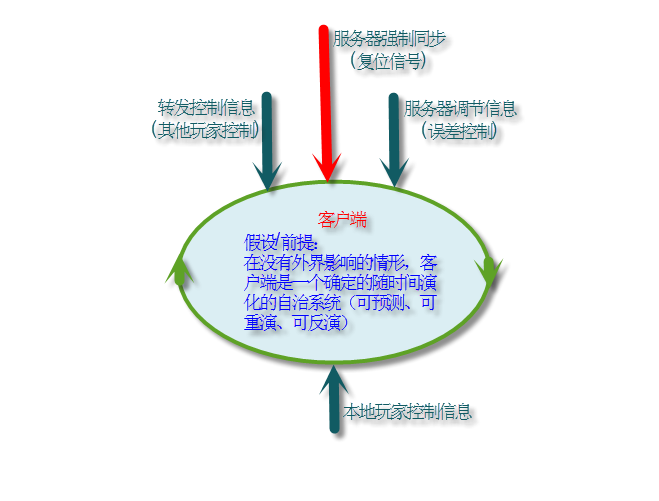


图3 从控制系统角度看客户端

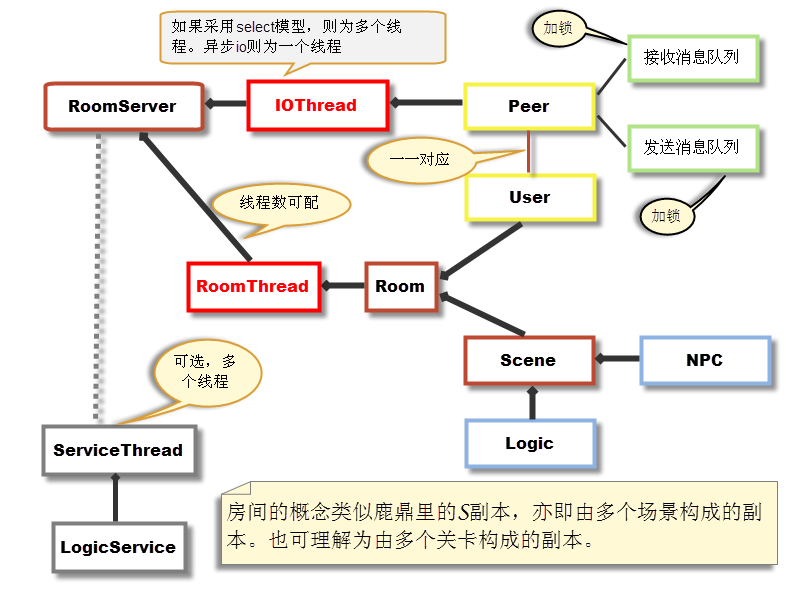


图4 服务器分级响应示意图

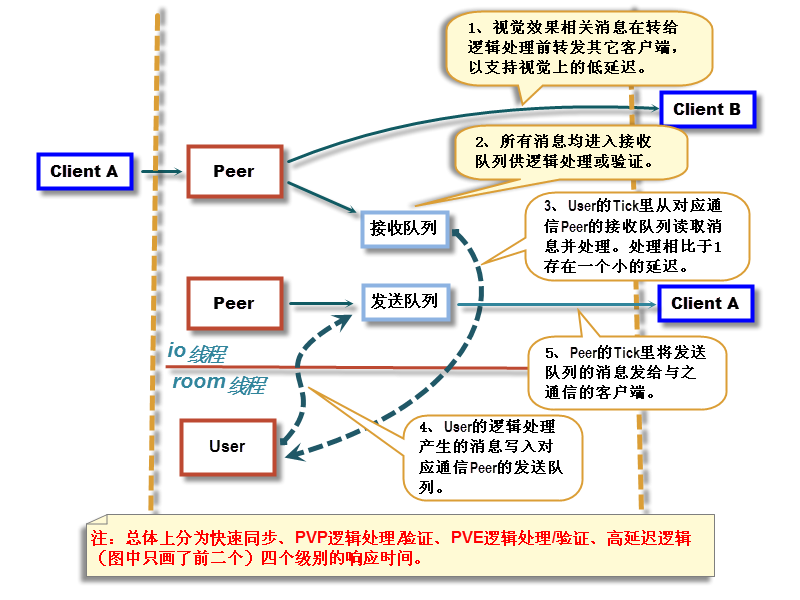


图5 分级响应总体设计示意

【关键点考虑】  
  
㈠与别的游戏的主要不同  
1、服务器（实现M与C）与客户端（实现M、V与C）近似同构，所以服务器会变简单，客户端会变复杂  
2、接近完全的分布式并行计算，理论上客户端实时性会更好，服务器战斗负载不会因在线人数指数增长（也因为分布式计算限制了更复杂的多人耦合）  
3、拥抱web，理论上将从web体系继承其异构系统整合能力、internet级的伸缩性与优秀的负载均衡  
4、相对而言客户端更复杂、更具自主性，需要确保正确性与安全性  
  
㈡高负载与低延时分离  
1、游戏里对负载（同服同时在线人数）的需求与实时性（战斗与战场）的需求大多是分离的。  
2、高负载：采用web架构。可以考虑chrome/html5+nginx+node.js+ejabberd。  
聊天服务器ejabberd自己支持集群。web前端可以用nginx做反向代理实现负载均衡（在nginx前还可以用dns做一级硬件均衡）。  
3、低延时：采取转发、模拟、协同与验证的方式，基本上是结合LOL与DNF/1018的方式。  
4、客户端采用2条通信通路分别连接高负载服务器与低延时服务器  
  
㈢转发与并行模拟  
1、转发  
是直接p2p还是p2sp？公司研发网络环境所限，使用p2p可能会面临比较大的调试与提前验证问题，如果转发量不是特别大，p2sp可能是个比较合适的选择。  
哪些内容需要转发？人机交互是最小转发集合，估计也是完备的转发集合。  
2、模拟  
几乎所有游戏服务器都会涉及模拟，这个方案要模拟的内容比别的游戏更多。这块的关键点：  
\*时间同步与延迟校准  
需要一种机制保证客户端与服务器的计时一致。  
客户端时间＝服务器时间-该客户端延迟  
\*基于时间戳的同步  
同步的消息必须是某种可预测行为的触发（所以可以实现模拟）。  
不进行消息包回溯修正，模拟的正确性的临界条件是接收到消息的延迟不能超过相邻2条消息的触发间隔。  
\*如果延迟做不到上一条，则需要记录消息历史并回溯修正（客户端行为），或者由服务器定时进行强制同步。  
\*客户端与服务器均需进行同一系统模拟，真实世界以服务器即时看到的为准（亦即服务器忽略各客户端的延迟），客户端模拟考虑相对服务器的延迟。由于CAP原理，这一选择不保证完全公平，但建立了一个什么是正确的标准。  
  
㈣服务器分级响应机制、不同级别延时不同  
1、消息转发  
2、快速模拟，用于pvp逻辑  
3、普通模拟，用于pve逻辑  
4、延迟模拟，用于无需频繁同步的逻辑，AI可能在这里

见图4与图5  
  
㈤分布式独立AI与战斗逻辑

每个客户端都采用本地逻辑行为，主要要考虑的是如何保证这些行为是确定的（亦即虽然不同的客户端都在并行自主运算，但结果是相同的）  
1、确定的随机数生成算法  
2、随机数种子同步  
3、确定性AI或基于前述随机数的确定AI  
4、本地战斗逻辑（客户端包含多玩家逻辑，其中有一个由本地控制，其它由网络控制，图3）  
5、客户端按服务器数据进行调节，关键数据等服务器同步数据到达再生效（图3）  
  
㈥ pvp实时判定，pve延迟裁决与同步  
由于游戏客户端本质上是不可信的，所以游戏系统很难避免服务器端的验证，这里的关键点主要是  
1、验证内容（空间数据差异、属性数据差异）  
2、同步内容（空间数据、属性数据、 pvp数据、收益相关结果数据 ）  
3、同步方法（空间与属性数据自动调节，pvp与收益相关数据直接以服务器为准）  
  
㈦信用评价与惩罚机制  
这一条是针对游戏角色的，服务器端需要记录玩家的信用数据，并有一套算法进行评估，系统自动或由人工进行滞后处理与惩罚。  
  
【相关技术】  
  
1、底层通信系统（webjet有一套比较方便的服务器端通信系统）  
2、mono/dotnet/c#  
3、chrome/html5  
4、node.js  
5、nginx/lvm/haproxy  
6、ejabberd/openfire  
7、自动控制算法  
8、神经网络&遗传算法  
9、异构计算（cpu+gpu）  
  
【运维相关考虑】  
  
1、运维系统与游戏系统隔离，通过日志及os功能进行关联  
2、运维系统可以基于web系统构建（node.js可用于服务端，我们打算尝试，还没开始弄）

【适应快速迭代的开发策略】

采用快速迭代开发时，通常需要短周期并保持持续可体验，采用我们这种架构可以按如下优先顺序进行开发：

1、纯客户端功能，保证原型快速开发与相对较好的单机体验（画面、操控、响应）

2、转发功能，大厅服务器，同时客户端改造为支持多玩家（1个本地控制，其它网络控制）

3、逐渐重构战斗相关模块，添加服务端模拟与验证，强制同步（可考虑MVC模式，服务器与客户端共用M，分别写不同的V与C）

4、添加客户端自动调节，解决正确性相关问题

5、服务端存储、玩家数据相关，版本工具

6、铺量

7、运维相关工具与统计日志

8、上线与运营相关工作  
  
【为什么没有与玩家相关的内容】  
  
架构往往解决非功能的问题，而与玩家相关的主要是功能问题。如果说高负载、低延时与玩家无关的话。  
  
【可以用于哪些游戏】  
  
1、大厅+对战（目前要做的就是这种）  
2、传统街机风格游戏（多人pve，需要低延时）  
3、 mmorpg（野外、主城与副本开在新的Server即可，亦即都用低延时策略；world、好友、聊天等采用高负载策略）  
4、传统大厅房间类  
  
\*其实暂时没想到哪种游戏不能用这种架构:)  
\*理论上应不限于端游，页游与手游亦可