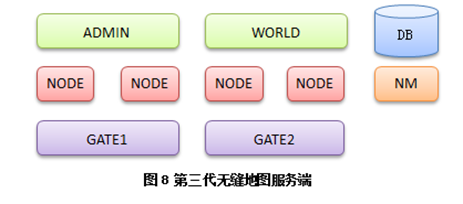
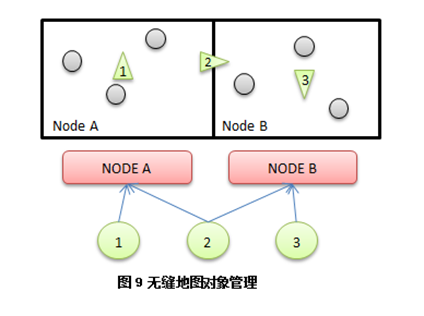
**类型4：第三代游戏服务器 2007**

从魔兽世界开始无缝世界地图已经深入人心，比较以往游戏玩家走个几步还需要切换场景，每次切换就要等待 LOADING个几十秒是一件十分破坏游戏体验的事情。于是对于 2005年以后的大型 MMORPG来说，无缝地图已成为一个标准配置。比较以往按照地图来切割游戏而言，无缝世界并不存在一块地图上面的人有且只由一台服务器处理了：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image28.png)

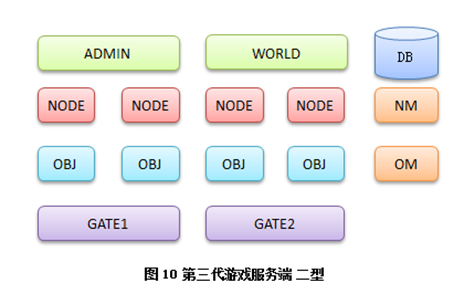
每台 Node服务器用来管理一块地图区域，由 NodeMaster（NM）来为他们提供总体管理。更高层次的 World则提供大陆级别的管理服务。这里省略若干细节服务器，比如传统数据库前端，登录服务器，日志和监控等，统统用 ADMIN概括。在这样的结构下，玩家从一块区域走向另外一块区域需要简单处理一下：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image29.png)

玩家1完全由节点A控制，玩家3完全由节点B控制。而处在两个节点边缘的2号玩家，则同时由A和B提供服务。玩家2从A移动到B的过程中，会同时向A请求左边的情况，并向B请求右边的情况。但是此时玩家2还是属于A管理。直到玩家2彻底离开AB边界很远，才彻底交由B管理。按照这样的逻辑将世界地图分割为一块一块的区域，交由不同的 Node去管理。

对于一个 Node所负责的区域，地理上没必要连接在一起，比如大陆的四周边缘部分和高山部分的区块人比较少，可以统一交给一个Node去管理，而这些区块在地理上并没有联系在一起的必要性。一个 Node到底管理哪些区块，可以根据游戏实时运行的负载情况，定时维护的时候进行更改 NodeMaster 上面的配置。

于是碰到第一个问题是很多 Node服务器需要和玩家进行通信，需要问管理服务器特定UID为多少的玩家到底在哪台 Gate上，以前按场景切割的服务器这个问题不大，问了一次以后就可以缓存起来了，但是现在服务器种类增加不少，玩家又会飘来飘去，按UID查找玩家比较麻烦；另外一方面 GATE需要动态根据坐标计算和哪些 Node通信，导致逻辑越来越厚，于是把：“用户对象”从负责连接管理的 GATE中切割出来势在必行于是有了下面的模型：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image31.png)

网关服务器再次退回到精简的网络转发功能，而用户逻辑则由按照 UID划分的 OBJ服务器来承担，GATE是按照网络接入时的负载来分布，而 OBJ则是按照资源的编号（UID）来分布，这样和一个用户通信直接根据 UID计算出 OBJ服务器编号发送数据即可。而新独立出来的 OBJ则提供了更多高层次的服务：

* 对象移动：管理具体玩家在不同的 Node所管辖的区域之间的移动，并同需要的 Node进行沟通。
* 数据广播：Node可以给每个用户设置若干 TAG，然后通知 Object Master 按照TAG广播。
* 对象消息：通用消息推送，给某个用户发送数据，直接告诉 OBJ，不需要直接和 GATE打交道。
* 好友聊天：角色之间聊天直接走 OBJ/OBJ MASTER。

整个服务器主体分为三层以后，NODE专注场景，OBJ专注玩家对象，GATE专注网络。这样的模型在无缝场景服务器中得到广泛的应用。但是随着时间的推移，负载问题也越来越明显，做个活动，远来不活跃的区域变得十分活跃，靠每周维护来调整还是比较笨重的，于是有了动态负载均衡。

动态负载均衡有两种方法，第一种是按照负载，由 Node Master 定时动态移动修改一下各个 Node的边界，而不同的玩家对象按照先前的方法从一台 Node上迁移到另外一台 Node上：

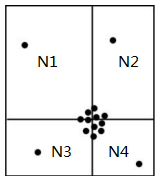
[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image33.png)

       图11 动态负载均衡

这样 Node Master定时查找地图上的热点区域，计算新的场景切割方式，然后告诉其他服务器开始调整，具体处理方式还是和上面对象跨越边界移动的方法一样。

但是上面这种方式实现相对复杂一些，于是人们设计出了更为简单直接的一种新方法：

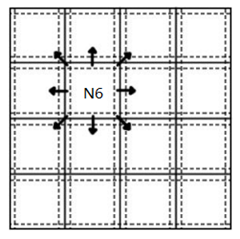
[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image30.png)

               图12 基于网格的动态负载均衡

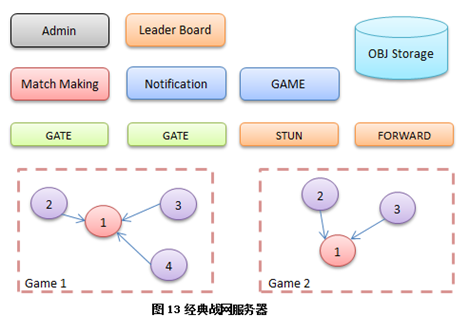
还是将地图按照标准尺寸均匀切割成静态的网格，每个格子由一个具体的Node负责，但是根据负载情况，能够实时的迁移到其他 Node上。在迁移分为三个阶段：准备，切换，完成。三个状态由Node Master负责维护。准备阶段新的 Node开始同步老 Node上面该网格的数据，完成后告诉NM；NM确认OK后同时通知新旧 Node完成切换。完成切换后，如果 Obj服务器还在和老的 Node进行通信，老的 Node将会对它进行纠正，得到纠正的 OBJ将修正自己的状态，和新的 Node进行通信。

很多无缝动态负载均衡的服务端宣称自己支持无限的人数，但不意味着 MMORPG游戏的人数上限真的可以无限扩充，因为这样的体系会受制于网络带宽和客户端性能。带宽决定了同一个区域最大广播上限，而客户端性能决定了同一个屏幕到底可以绘制多少个角色。

从无缝地图引入了分布式对象模型开始，已经完全脱离 MUDOS体系，成为一种新的服务端模型。又由于动态负载均衡的引入，让无缝服务器如虎添翼，容纳着超过上一代游戏服务器数倍的人数上限，并提供了更好的游戏体验，我们称其为第三代游戏服务端架构。网游以大型多人角色扮演为开端，RPG网游在相当长的时间里一度占据90%以上，使得基于 MMORPG的服务端架构得到了蓬勃的发展，然而随着玩家对RPG的疲惫，各种非MMORPG 游戏如雨后春笋般的出现在人们眼前，受到市场的欢迎。

**类型5：战网游戏服务器**

经典战网服务端和 RPG游戏有两个区别：RPG是分区分服的，北京区的用户和广州区的用户老死不相往来。而战网，虽然每局游戏一般都是 8人以内，但全国只有一套服务器，所有的玩家都可以在一起游戏，而玩家和玩家之使用 P2P的方式连接在一起，组成一局游戏：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image35.png)

玩家通过 Match Making 服务器使用：创建、加入、自动匹配、邀请 等方式组成一局游戏。服务器会选择一个人做 Host，其他人 P2P连接到做主的玩家上来。STUN是帮助玩家之间建立 P2P的牵引服务器，而由于 P2P联通情况大概只有 75%，实在联不通的玩家会通过 Forward进行转发。

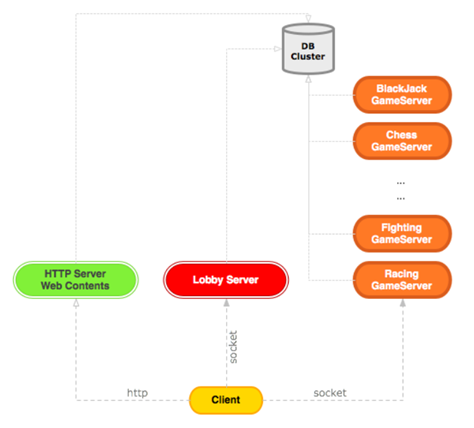
大量的连接对战，体育竞技游戏采用类似的结构。P2P有网状模型（所有玩家互相连接），和星状模型（所有玩家连接一个主玩家）。复杂的游戏状态在网状模型下难以形成一致，因此星状P2P模型经受住了历史的考验。除去游戏数据，支持语音的战网系统也会将所有人的语音数据发送到做主的那个玩家机器上，通过混音去重再编码的方式返回给所有用户。

战网类游戏，以竞技、体育、动作等类型的游戏为主，较慢节奏的 RPG（包括ARPG）有本质上的区别，而激烈的游戏过程必然带来到较 RPG复杂的多的同步策略，这样的同步机制往往带来的是很多游戏结果由客户端直接计算得出，那在到处都是破解的今天，如何保证游戏结果的公正呢？

主要方法就是投票法，所有客户端都会独立计算，然后传递给服务器。如果结果相同就更新记录，如果结果不一致，会采取类似投票的方式确定最终结果。同时记录本剧游戏的所有输入，在可能的情况下，找另外闲散的游戏客户端验算整局游戏是否为该结果。并且记录经常有作弊嫌疑的用户，供运营人员封号时参考。

**类型7：休闲游戏服务器**

休闲游戏同战网服务器类似，都是全区架构，不同的是有房间服务器，还有具体的游戏服务器，游戏主体不再以玩家 P2P进行，而是连接到专门的游戏服务器处理：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image36.png)

和战网一样的全区架构，用户数据不能象分区的 RPG那样一次性load到内存，然后在内存里面直接修改。全区架构下，为了应对一个用户同时玩几个游戏，用户数据需要区分基本数据和不同的游戏数据，而游戏数据又需要区分积分数据、和文档数据。胜平负之类的积分可以直接提交增量修改，而更为普遍的文档类数据则需要提供读写令牌，写令牌只有一块，读令牌有很多块。同帐号同一个游戏同时在两台电脑上玩时，最先开始的那个游戏获得写令牌，可以操作任意的用户数据。而后开始的那个游戏除了可以提交胜平负积分的增量改变外，对用户数据采用只读的方式，保证游戏能运行下去，但是会提示用户，游戏数据锁定。

**类型8：现代动作网游**

从早期的韩国动作游戏开始，传统的战网动作类游戏和 RPG游戏开始尝试融合。单纯的动作游戏玩家容易疲倦，留存也没有 RPG那么高；而单纯 RPG战斗却又慢节奏的乏味，无法满足很多玩家激烈对抗的期望，于是二者开始融合成为新一代的：动作 + 城镇 模式。玩家在城镇中聚集，然后以开副本的方式几个人出去以动作游戏的玩法来完成各种 RPG任务。本质就是一套 RPG服务端+副本服务端。由于每次副本时人物可以控制在8人以内，因此可以获得更为实时的游戏体验，让玩家玩的更加爽快。

说了那么多的游戏服务器类型，其实也差不多了，剩下的类型大家拼凑一下其实也就是这个样子而已。游戏服务端经历了那么多结构上的变迁，内部开发模式是否依然不变？究竟是继续延续传统的开发方式？还是有了更多突破性的方法？经历那么多次架构变迁，后面是否有共通的逻辑？未来的发展还会存在哪些困难？游戏服务端开发如何达到最终的彼岸？请看下节：技术的演进。