手游页游和端游，本质上没有区别，区别的是游戏类型：

《游戏服务端架构发展史》转载请著名出处：<http://www.skywind.me/blog/archives/1265>

**类型1：卡牌，跑酷等弱交互服务端**

卡牌跑酷类因为交互弱，玩家和玩家之间不需要实时面对面PK，打一下对方的离线数据，计算下排行榜，买卖下道具即可，所以实现往往使用简单的 HTTP服务器：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image18.png)

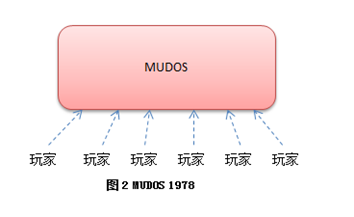
登录时可以使用非对称加密（RSA, DH），服务器根据客户端uid，当前时间戳还有服务端私钥，计算哈希得到的加密 key 并发送给客户端。之后双方都用 HTTP通信，并用那个key进行RC4加密。客户端收到key和时间戳后保存在内存，用于之后通信，服务端不需要保存 key，因为每次都可以根据客户端传上来的 uid 和 时间戳 以及服务端自己的私钥计算得到。用模仿 TLS的行为，来保证多次 HTTP请求间的客户端身份，并通过时间戳保证同一人两次登录密钥不同。

每局开始时，访问一下，请求一下关卡数据，玩完了又提交一下，验算一下是否合法，获得什么奖励，数据库用单台 MySQL或者 MongoDB即可，后端的 Redis做缓存（可选）。如果要实现通知，那么让客户端定时15秒轮询一下服务器，如果有消息就取下来，如果没消息可以逐步放长轮询时间，比如30秒；如果有消息，就缩短轮询时间到10秒，5秒，即便两人聊天，延迟也能自适应。

此类服务器用来实现一款三国类策略或者卡牌及酷跑的游戏已经绰绰有余，这类游戏因为逻辑简单，玩家之间交互不强，使用 HTTP来开发的话，开发速度快，调试只需要一个浏览器就可以把逻辑调试清楚了。

**类型2：第一代游戏服务器 1978**

1978年，英国著名的财经学校University of Essex的学生 Roy Trubshaw编写了世界上第一个MUD程序《MUD1》，在University of Essex于1980年接入 ARPANET之后加入了不少外部的玩家，甚至包括国外的玩家。《MUD1》程序的源代码在 ARPANET共享之后出现了众多的改编版本，至此MUD才在全世界广泛流行起来。不断完善的 MUD1的基础上产生了开源的 MudOS（1991），成为众多网游的鼻祖：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image19.png)

MUDOS采用 C语言开发，因为玩家和玩家之间有比较强的交互（聊天，交易，PK），MUDOS使用单线程无阻塞套接字来服务所有玩家，所有玩家的请求都发到同一个线程去处理，主线程每隔1秒钟更新一次所有对象（网络收发，更新对象状态机，处理超时，刷新地图，刷新NPC）。

游戏世界采用房间的形式组织起来，每个房间有东南西北四个方向可以移动到下一个房间，由于欧美最早的网游都是地牢迷宫形式的，因此场景的基本单位被成为 “房间”。MUDOS使用一门称为LPC的脚本语言来描述整个世界（包括房间拓扑，配置，NPC，以及各种剧情）。游戏里面的高级玩家（巫师），可以不断的通过修改脚本来为游戏添加房间以及增加剧情。早年 MUD1上线时只有17个房间，Roy Trubshaw毕业以后交给他的师弟 Richard Battle，在 Richard Battle手上，不断的添加各种玩法到一百多个房间，终于让 MUD发扬光大。

用户使用 Telnet之类的客户端用 Tcp协议连接到 MUDOS上，使用纯文字进行游戏，每条指令用回车进行分割。比如 1995年国内第一款 MUD游戏《侠客行》，你敲入："go east"，游戏就会提示你：“后花园 – 这里是归云庄的后花园，种满了花草，几个庄丁正在浇花。此地乃是含羞草生长之地。这里唯一的出口是 north。这里有：花待 阿牧（A mu），还有二位庄丁（Zhuang Ding）”，然后你继续用文字操作，查看阿牧的信息：“look a mu”，系统提示：“花待 阿牧（A mu）他是陆乘风的弟子，受命在此看管含羞草。他看起来三十多岁，生得眉清目秀，端正大方，一表人才。他的武艺看上去【不是很高】，出手似乎【极轻】”。然后你可以选择击败他获得含羞草，但是你吃了含羞草却又可能会中毒死亡。在早期网上资源贫乏的时候，这样的游戏有很强的代入感。

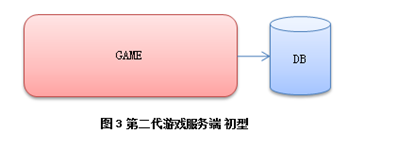
用户数据保存在文件中，每个用户登录时，从文本文件里把用户的数据全部加载进来，操作全部在内存里面进行，无需马上刷回磁盘。用户退出了，或者每隔5分钟检查到数据改动了，都会保存会磁盘。这样的系统在当时每台服务器承载个4000人同时游戏，不是特别大的问题。从1991年的 MUDOS发布后，全球各地都在为他改进，扩充，退出新版本，随着 Windows图形机能的增强。1997游戏《UO》在 MUDOS的基础上为角色增加的x,y坐标，为每个房间增加了地图，并且为每个角色增加了动画，形成了第一代的图形网络游戏。

因为游戏内容基本可以通过 LPC脚本进行定制，所以MUDOS也成为名副其实的第一款服务端引擎，引擎一次性开发出来，然后制作不同游戏内容。后续国内的《万王之王》等游戏，很多都是跟《UO》一样，直接在 MUDOS上进行二次开发，加入房间的地图还有角色的坐标等要素，该架构一直为国内的第一代 MMORPG提供了稳固的支持，直到 2003年，还有游戏基于 MUDOS开发。

虽然后面图形化增加了很多东西，但是这些MMORPG后端的本质还是 MUDOS。随着游戏内容的越来越复杂，架构变得越来越吃不消了，各种负载问题慢慢浮上水面，于是有了我们的第二代游戏服务器。

**类型3：第二代游戏服务器 2003**

2000年后，网游已经脱离最初的文字MUD，进入全面图形化年代。最先承受不住的其实是很多小文件，用户上下线，频繁的读取写入用户数据，导致负载越来越大。随着在线人数的增加和游戏数据的增加，服务器变得不抗重负。同时早期 EXT磁盘分区比较脆弱，稍微停电，容易发生大面积数据丢失。因此第一步就是拆分文件存储到数据库去：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image20.png)

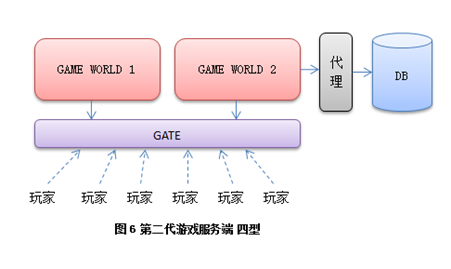
此时游戏服务端已经脱离陈旧的 MUDOS体系，各个公司在参考 MUDOS结构的情况下，开始自己用 C在重新开发自己的游戏服务端。并且脚本也抛弃了 LPC，采用扩展性更好的 Python或者 Lua来代替。由于主逻辑使用单线程模型，随着游戏内容的增加，传统单服务器的结构进一步成为瓶颈。于是有人开始拆分游戏世界，变为下面的模型：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image21.png)

游戏服务器压力拆分后得意缓解，但是两台游戏服务器同时访问数据库，大量重复访问，大量数据交换，使得数据库成为下一个瓶颈。于是形成了数据库前端代理（DB Proxy），游戏服务器不直接访问数据库而是访问代理，再有代理访问数据库，同时提供内存级别的cache。早年 MySQL4之前没有提供存储过程，这个前端代理一般和 MySQL跑在同一台上，它转化游戏服务器发过来的高级数据操作指令，拆分成具体的数据库操作，一定程度上代替了存储过程：

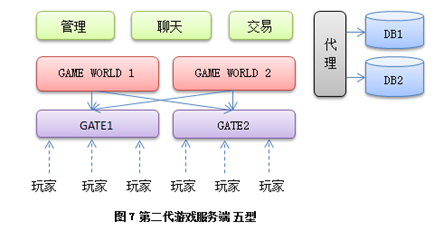
但是这样的结构并没有持续太长时间，因为玩家切换场景经常要切换连接，中间的状态容易错乱。而且游戏服务器多了以后，相互之间数据交互又会变得比较麻烦，于是人们拆分了网络功能，独立出一个网关服务 Gate（有的地方叫 Session，有的地方叫 LinkSvr之类的，名字不同而已）：[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image22.png)

但是这样的结构并没有持续太长时间，因为玩家切换场景经常要切换连接，中间的状态容易错乱。而且游戏服务器多了以后，相互之间数据交互又会变得比较麻烦，于是人们拆分了网络功能，独立出一个网关服务 Gate（有的地方叫 Session，有的地方叫 LinkSvr之类的，名字不同而已）：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image24.png)

把网络功能单独提取出来，让用户统一去连接一个网关服务器，再有网关服务器转发数据到后端游戏服务器。而游戏服务器之间数据交换也统一连接到网管进行交换。这样类型的服务器基本能稳定的为玩家提供游戏服务，一台网关服务1-2万人，后面的游戏服务器每台服务5k-1w，依游戏类型和复杂度不同而已，图中隐藏了很多不重要的服务器，如登录和管理。这是目前应用最广的一个模型，到今天任然很多新项目会才用这样的结构来搭建。

人都是有惯性的，按照先前的经验，似乎把 MUDOS拆分的越开性能越好。于是大家继续想，网关可以拆分呀，基础服务如聊天交易，可以拆分呀，还可以提供web接口，数据库可以拆分呀，于是有了下面的模型：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image27.png)

这样的模型好用么？确实有成功游戏使用类似这样的架构，并且发挥了它的性能优势，比如一些大型 MMORPG。但是有两个挑战：每增加一级服务器，状态机复杂度可能会翻倍，导致研发和找bug的成本上升；并且对开发组挑战比较大，一旦项目时间吃紧，开发人员经验不足，很容易弄挂。

比如我见过某上海一线游戏公司的一个 RPG上来就要上这样的架构，我看了下他们团队成员的经验，问了下他们的上线日期，劝他们用前面稍微简单一点的模型。人家自信得很，认为有成功项目是这么做的，他们也要这么做，自己很想实现一套。于是他们义无反顾的开始编码，项目做了一年多，然后，就没有然后了。

现今在游戏成功率不高的情况下，一开始上一套比较复杂的架构需要考虑投资回报率，比如你的游戏上线半年内 PCU会去到多少？如果一个 APRG游戏，每组服务器5千人都到不了的话，那么选择一套更为贴近实际情况的结构更为经济。即使后面你的项目真的超过5千人朝着1万人目标奔的话，相信那个时候你的项目已经挣大钱了 ，你数着钱加着班去逐步迭代，一次次拆分它，相信心里也是乐开花的。

上面这些类型基本都是从拆分 MUDOS开始，将 MUDOS中的各个部件从单机一步步拆成分布式。虽然今天任然很多新项目在用上面某一种类似的结构。因为他们本质上都是对 MUDOS的分解，故将他们归纳为第二代游戏服务器。