浅谈分布式系统—曾经的 12306 购票网站

一.系统介绍和系统架构

12306 网站是中国铁路系统的官方售票网站，于 2011 年 6 月 12 日投入运营，2011 年年底，全国铁路已经全面推行网络售票，中国铁路开始走进电子商务时代。

作为一个数据流量非常大的分布式系统，12306 网站系统整体按分层架构处理，每一层都是可注册、可插拔的体系，这种架构的好处是每一层都可以分层优化，而互不影响。并能根据实际运行的情况对并发和访问量过大的实体层进行动态扩容，很容易提高系统的并发和稳定性。

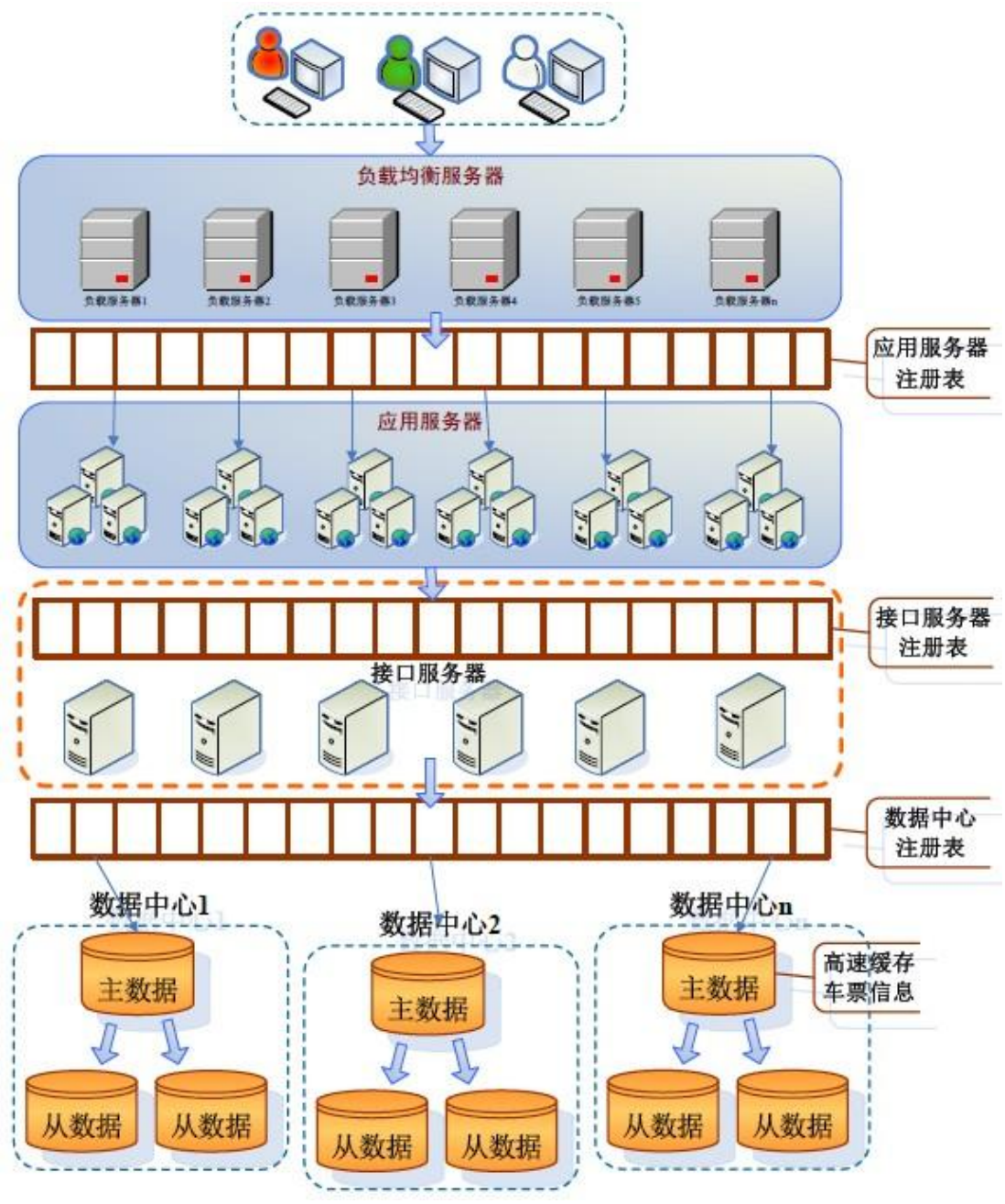

图 1：服务器架构图[1]

此架构很好地解决了应用服务器和数据访问的瓶颈问题，如果应用服务器压力大则可以通过注册表对应用服务器扩容，并通过负载均衡均衡地访问各个应用服务器。如果数据访问是一个瓶颈则可增加数据加心的方式来解决数据访问拥挤情况。[1]

硬件方面，初期 12306 用的是 HP Superdome 小型机，UNIX 系统，Sybase ASE 数据库，发现性能严重不足，遂改用 x86 系统+linux 平台。最后核心系统用了十七个节点的多路 Xeon E7，每个节点配 1TB 内存，数据库全部在内存中运行。[2]

二.系统面临的挑战

由于 12306 网站是唯一一个法定的火车票销售网站，在春节前后将会面对极大的访问量，所以对于此系统来讲，可扩展性是其最为重要和最为迫切的需求。另外作为公开的涉及金钱和个人信息的网站，安全性也是非常重要的一部分。

在访问量方面，网络数据表明，在春运峰值时刻，单日用户量可达千万级。巨大的用户数量在每天固定的几个时刻集中抢购车票，给服务器造成了巨大的压力。工作人员先后几次升级系统软硬件，但是仅仅起到了一定的缓解作用：2012 年初，网站出现了整体性的崩溃现象；2013 年春运，网站虽然仍然严重拥塞，但始终保持可用；2014 年春运，问题仍然以拥塞现象为主，但是必须考虑到各种抢票软件的出现，服务器承载的计算量必然比以往更多。另外，出现了程序漏洞导致个人信息泄露的问题。

在服务器计算能力方面，12306 采用了分布式内存数据平台技术。使用较少的大型服务器，将需要频繁访问的数据全部放在内存中运行，由于内存访问速度（100ns）比硬盘（10ms）要快 5 个数量级，因此能够提升短时间内的并发处理数据量。[3]

然而，这种现在很先进的技术并未能够真正解决问题，在高峰时段仍然出现极大的延迟以致用户无法正常购票。由于网站在平时运行的很正常，可以得知：导致各种问题出现的根本原因就是过大的并发访问量超过了服务器的承受量。继续增加服务器的计算能力，是很困难的，受到现在科学发展水平的制约，而减少总访问量是可行的。高峰时段旅客的需求是不会减少的，因此要想减少访问量，应该从减少旅客每一次操作带来的网络数据流量和服务器计算量来入手。

目前，很多用户对网站产生了不满。主要集中在两个方面：第一：没能买到票。第二：虽然买到了票，但是操作过程十分辛苦，浪费了大量时间。

春运期间，火车票是紧俏商品，供小于求的现状不是程序员能够改变的，所以必然有人买不到票。但是，程序员应该做到的是：第一：让旅客花费比较少的时间精力去买票，第二：不受网络延迟和操作的困扰，第三：把票以公平的方式卖给真正的旅客而不是倒卖车票的黄牛。

现有系统的第一个严重问题，就是存在大量的冗余数据流。我去年出差去上海，想要买一张高铁二等座票，操作流程是什么呢？登录，输入目的地和出发地，系统马上输出了当天北京到上海的全部 40 个车次及座位余票量，我只想坐高铁，选择车次类型后，网站刷新出了 33 个高铁车次。然后我要把出发日期改到一周之后，系统又一次返回了那一天的 33 个车次余票量。我想要 9 点出发，再次选择发车时间在上午，系统重新刷新了 15 个上午出发的车次。我最终选择了一个车次（G1）来购买。

在我这个操作流程中，如果每个车次每种座位余票量需要 1 个字节来传输，我一共从服务器获得了（40+33+33+15）*11（席别）共 1331 字节的数据，而实际上起了作用的最多也就是出发日 9 点附近 5 个车次二等座余票量的 5 个字节，其它 1000 多个字节全部为冗余信息，浪费率高达 99.6%。同时每次刷新还传递了图片文字等非必要数据，进一步增加了无效的网络流量。

同时，我的每一次查询不仅使用了网络资源，还要让服务器进行一些计算。北京到上海是非常常见的直达线路，直接输出对应车次即可。但是很多人的目的地是中小城市，问题就会复杂。例如搜索广州到呼和浩特的车次，系统在计算后输出无直达车次的提示。选择自动中转查询，系统计算后仍然无输出结果。手动输入中转站北京，系统输出了北京-广州和北京-呼和浩特的相应车次。这些搜索，需要服务器进行计算，在高峰时段，又耗费了宝贵的系统资源。

现有系统的第二个严重问题是：关键数据与非关键数据混行。12306 网站对于用户来讲就是一个网页，用户的一切操作本质上是通过浏览器发送数据包到 www.12306.cn，服务器后台再根据数据包中的内容安排系统内分工。在高峰时段，用户都在关心车票余量，以致不断进行刷新（不管是手动还是自动），导致数据量极大，负责查询的内部网络拥塞。用户发现拥塞之后，最常见的手段还是刷新，大量不断刷新的查询请求进一步导致了更大拥塞，这种拥塞促使更多的用户进行刷新，产生恶性循环。最终，由于所有的查询数据包都积压在系统入口处，导致交易，身份验证等环节的数据也无法正常流通。因为服务器在处理一个数据包之前无法知道它是查询包还是交易包。本来每人都进行一个查询或者一个交易操作，系统入口都不拥塞，但是若每人都积压了多次查询，系统入口就要拥塞了，负责交易的后台却在闲置，因为交易数据包和查询数据包混合在了一起，拥塞在系统入口处。

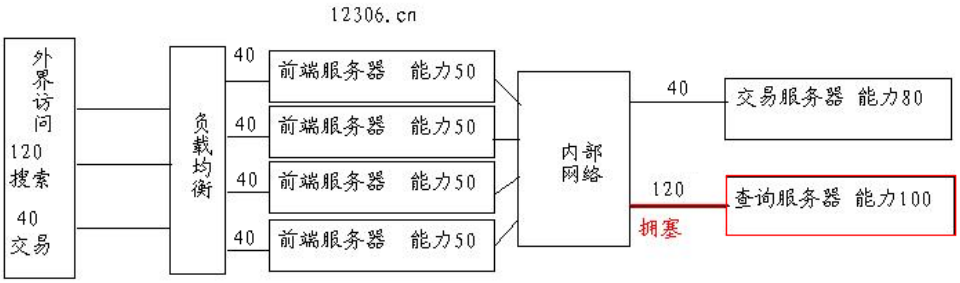

图 2.1 本来只有查询服务器拥塞

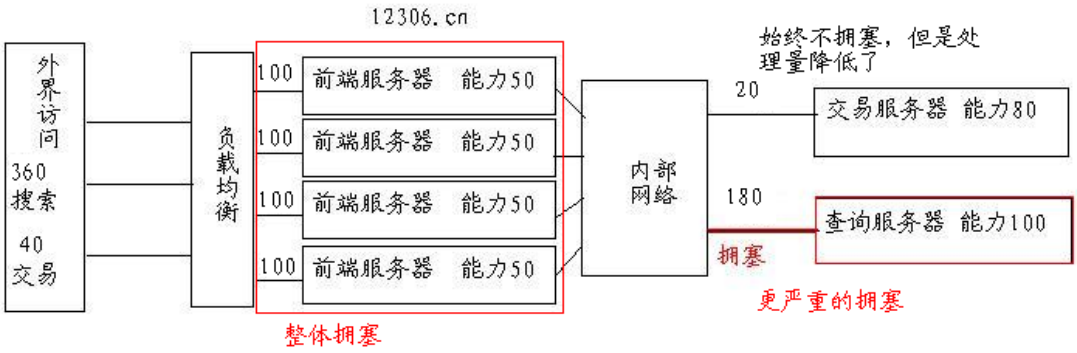

图 2.2 恶性循环导致整体拥塞

图 2：系统拥塞示意图

现有系统的第三个问题是：无法从根本上杜绝抢票软件的存在。网站的本质还是一个网页，客户和服务器之间的交流是一个浏览器封装的数据包，各大浏览器都是非常普及的，已经被研究的非常透彻，程序员想要写一个程序冒充浏览器发送数据是非常容易的。这种情况的存在，轻则一台电脑多开数个浏览器窗口抢票，重则出现专业程序自动大量产生购票请求。这种情况现在已经发生，未能得到有效遏制，导致本来不想使用这些手段的旅客不得不去使用，否则根本不可能买到车票，形成恶性循环：本来 3000 人抢 2000 张票，每人一个进程，服务器处理 3000 个请求；现在还是 3000 人抢 2000 张票，每人 5 个进程，服务器就得处理 15000 个请求，加剧了拥塞，而买到票的人数绝对不会增加。

第四个问题是公平性：火车票是公共资源，在春运期间是稀缺资源，应该以公平的手段分给竞争者。网络售票出现之前，大家彻夜排队，是一种相对的公平：排在前面的人付出了更多时间，买到票的可能性就大。现在的网络购票机制，则完全没能体现公平性，整点放票的时候所有旅客一起刷网页，谁先进入购票系统纯属偶然，并没有人付出更多：先来后到这个分配方法根本就不适用于网络抢购车票这个环境，抢票靠先点击这个分配方法从指导思想上就是落后的和错误的。现行的售票机制，如同现实中在站前广场外设一个栏杆，10 点打开，外面的人一起拥挤过来“排队”。

第五个问题是防范黄牛：售票系统数次升级，使用了让大家眼花缭乱的验证码，却未能有效地打击黄牛。这又是一个错误的思想导致的问题：系统越复杂，专业人员和业余人员的差距就越大。黄牛都是专业的，正常的旅客绝大部分是业余的。就算黄牛没有任何可用的自动工具，手动操作的熟练度也超过绝大部分正常旅客，系统越复杂，操作越麻烦，技巧越多，黄牛越高兴。

三.我的观点和建议

12306 售票系统面临的问题是巨大的访问量，以前的各种改进方法，都把重点放在了网站自身的建设上面，但是一个网站，一个服务器群组要处理千万级的客户，计算能力（网络带宽）将始终是其瓶颈。为了解决这个问题，我认为，应该放开视野，不要只看到网站内部的计算能力，而让千万级的客户机一起参与其力所能及的计算，就可以大大的缓解网站的压力。

在整个购票过程中，只有售票这一个环节存在真正的并发性，客户间的操作必须具有互斥性，方能避免一张票卖给多人情况的发生。在查询阶段和付款阶段，并没有互斥性存在，这些工作应该和售票工作隔离开，并尽可能不和售票工作争抢系统资源。

3.1 推出客户端

为了尽可能调动客户机的计算能力，我认为，12306 网站应该推出独立的客户端，网站仅作为展示环节，包括人工客户服务，客户端下载等工作。而全部的查询、售票、交易工作都由客户端来进行。

使用客户端有以下几个好处：

第一：节约了高峰时段的网络流量和服务器计算资源。例如列车时刻表等固定不变的信息可以预先保存在客户端的数据库中。当用户搜索车次的时候，客户端根本不需要向服务器传递任何信息，只在用户确定车次想要购票时才向服务器请求数据。同样进行前面问题一中的两个搜索，我购买北京到上海的车票，在客户端输入我的需求（一周后，指定时间的车次，二等座余票量），客户端向查询服务器请求 5 个数据，马上返回，我进入购票阶段，没有浪费网络资源。有人搜索广州到呼和浩特，虽然找寻路线有一定复杂性，但是常规配置的个人计算机很容易就能根据客户端内置的时刻表完成各大车站的遍历搜索，并给出换乘方案。在这个计算上，服务器并不比个人电脑做得更好。然后客户端向服务器请求其指定车次的余票量，不使用服务器的计算能力。

第二：方便的将各种类型的数据隔离开，一种数据的拥塞不会对其他数据造成影响。客户端可以内置数个不同的网址信息，当登录时访问网址一，网址一只包含身份验证服务器；进行查询时向网址二发送请求，网址二只包含查询服务器；以此类推。由于在最开始就把各种数据分开了，现在不同数据各行其道，所有的服务器都能全速投入工作而不受到干扰。

第三：有效地防范了各种第三方软件的存在。客户端由网站自制，内部可以加入密钥等机制，发送的数据较难由其他软件模仿。从根本上杜绝了抢票软件的出现。这样，不仅难以制作抢票软件，制作抢票软件者还要因为必须破译客户端内的密钥而担受更大的法律风险。

而客户端可以简单地进行密钥更新，使得抢票软件马上失效。这样客户都会选择标准的客户端来购票，不仅简单，而且保证公平。

第四：通过比赛手速和网速来分配车票是一种非常不合理的方法，应该完全杜绝掉。例如今天十点钟发售一种车票，从 9:55 到 10:05 都可以下单，每个客户端向服务器发送一个数据包，里面包含旅客想要的车次、席别、身份标识。服务器接受这些数据包，不记录其到达时间，预先排除不合法申请。10:05 时统一按照随机算法排序，依次满足需求，分别返回购票结果。由于取消了时间排序，极大地缓解了尖峰时刻的出现。相比旧的方法，旅客不需要着急，提交需求后静等结果即可。铁路方面，15 分钟就将这个车次的最大抢票高峰化解，之后可以恢复经典的销售方法，不会再有高峰出现，先到先得的方法可以正常运作了。

第五：客户端可以帮助打击黄牛，消除黄牛在抢票能力上的优势。客户端本身是一个独立的程序，可以通过一些技术手段达到独占运行，一台计算机只能使用一个客户端，增加了黄牛抢票的困难。客户端上的操作简单明了，所有人来操作效果都等价，确保了公平性。

第六：客户端操作过程不受网络性能影响，方便迅速，即使服务器发生拥塞，用户也能将受到的影响降至最低，仍旧可以进行不需要服务器参与的工作。客户端可以很容易的进行自定义设置，例如预先保存出行计划（我经常来往北京和上海，这是默认页面），购票时选择优先级（最好是硬卧，买不到就买硬座），等等。给用户最好的操作体验。

第七：有较好的可扩展性，现在手机等非传统上网渠道非常流行，网站要适应手机需要修改很多，每次更新都要考虑适应性。而客户端则可以为手机单独编写，手机和电脑客户端不同，但是传送的数据相同，客户端的差异对服务器端透明，有利于维护。

3.2 服务器端的设计

作为服务器端，首先要保证安全性与稳定性；其次，不能改变网络售票与传统售票系统间的接口。

作为一个工作压力很大的服务器，我们已经通过客户端分流走了大部分计算需求，现在服务器只需要进行以下工作：

1.发布并更新客户端

这是一项静态服务，包括提供客户端的下载，以及客户端数据部分的更新检查（例如开通了临时列车）。由于这些内容变化较少，没有实时性，使用常规的下载服务器管理即可，不会产生问题。

2.进行身份验证

当一个客户端登录系统，需要先进行身份验证。服务器验证账号和密码，读取此账号的信息，并给出相应的密钥来标示此用户。其它服务器只能看到代表用户身份的加密信息，姓名和身份证号等个人隐私将不再出现在外网上，保证了用户的个人隐私。本服务器还可以与公安机关等机构连接，进行实名制查验等工作。

3.车票余量查询

用户通过客户端选择了车次和席别，并通过客户端发送了查询请求。查询服务器将全部的车票余量放置在内存中，以便尽快响应请求。查询服务器和售票服务器是相互独立的，查询服务器作为唯一的客户每隔一个短时间向售票服务器请求全部车次真正的余票量，用这个数据在短时间内响应外界的全部请求。

4.售票服务器

售票服务器是系统的核心，应该由高性能群组来构成。前端计算机按照客户请求的车次将数据分发给对应的后端计算机。后端一部分计算机接收客户端发来的高峰时段购票请求，按照随机排序等方法将高峰数据转化成已经完成排队的数据流；另一部分计算机通过简单的排队算法处理这个数据流和非高峰数据。

5.付款服务器

付款服务器并不需要处理尖峰数据，可以对并发的请求进行简单排队。客户不会在乎付款前是否等待了几秒钟，因为他们已经买到票了。

6.内部接口服务器

内部接口服务器用来联通网络售票系统和传统售票系统，将网络售票成功售出的车票信息输入传统售票系统，同时接受传统售票系统给出的可售车票信息。本服务器不与外界连接，计算工作主要是将身份验证服务器给出的身份-密钥关系和付款服务器给出的身份密钥-车票信息重新解密组合，使车票真正完成销售。票务信息传递到了传统售票系统中，旅客可以取票乘车了。同时，此服务器也可以担负依据身份信息鉴别不合法车票的工作。

7.网络结构和数据流

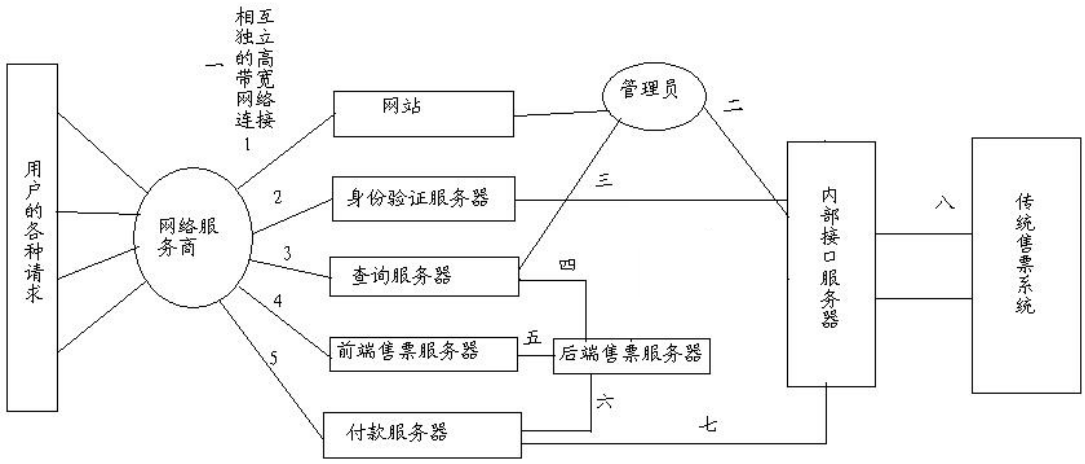

图 3　我设计的系统架构

数据流一，是客户到售票系统的总数据流。售票系统五个对外服务的模块各自有不同的网络地址，使用相互独立的网络连接接入因特网。这样，任何一个模块发生拥塞，不会影响其他模块的正常工作，各种不同请求的分类数据包本来由网站前端依据其中的数据计算后处理，现在在因特网中就依据不同的地址而自动的实现了分流。各个模块的前端可以按照需求加入负载平衡等模块，但是依旧是相互独立的。

数据流 1，用户访问网站，下载或更新客户端。

数据流 2，用户以用户名登陆，获取密钥，之后可以传送加密的身份信息。

数据流 3，用户通过客户端直接发送包括（日期-车次-席别）的查询请求，通过在客户端上的选择，只请求用户真正关心的数据，减少不必要的数据量。

数据流 4，用户发送包括（加密身份信息-日期-车次-席别）的购票请求，在后端服务器给出购票结果后，服务器返回成功购票的（加密身份信息-日期-车次-席别-编号）信息。

数据流 5，用户依据售票服务器给出的购票成功编号进行付款。

数据流二，内部接口服务器提供车次的变化等不常变化的数据，管理员据此更新客户端，刷新可售票量等信息。这些信息不常变化，甚至可以手动完成。

数据流三，身份验证服务器将身份密钥对应的真实身份信息数据传递给内部接口服务器，以便用户完成购票后生成有效车票。

数据流四，查询服务器需要定时和后端售票服务器同步票量数据。

数据流五，前端售票服务器把高峰期的集中购票请求按照随机方式整理成顺序到来的请求。后端售票服务器返回购票结果。

数据流六，后端售票服务器将可以满足的购票需求传递给付款服务器，用户可以付款买票了。

数据流七，用户付款后，付款服务器将完成付款的车票信息传递给内部接口服务器。

数据流八，内部接口服务器传递车票和用户信息给传统售票系统。

四.总结

我所设计的系统，作为一个分布式系统，将透明性工作放在客户端中处理，系统对于客户机并不透明，但是对于用户是透明的。这个思路可以充分利用客户机的计算能力，和整个因特网的路由能力，分流了计算量，有利于在高峰时段保持系统正常工作。

我的系统中各个模块，每一个都可以作为一个小的分布式系统来建造。面对巨大访问量带来的困境，我没有能力直接通过提高计算能力去面对，但是有能力去减少和分流访问量，逐步消除这些困难。

这篇文章的最初构思，来自网页[2]中网友的争辩，他们的意见给了我相当大的启发。

参考网页：

【1】http://www.uml.org.cn/yunjisuan/201303266.asp

【2】http://www.zhihu.com/question/22451397

【3】http://server.chinabyte.com/151/12820151.shtml