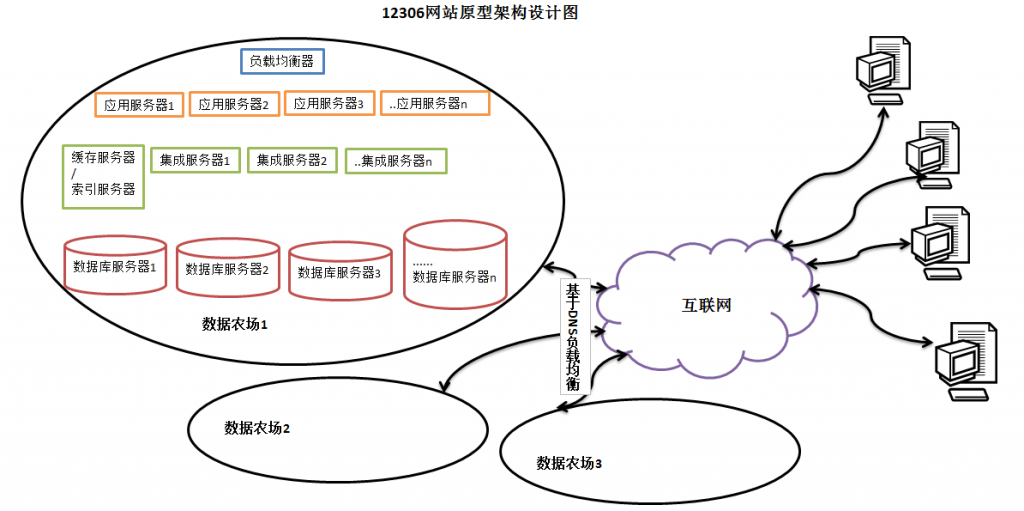
[假如我来架构12306网站(一) - 概论](http://blog.csdn.net/jackxinxu2100/article/details/8140755)

**序言：  此文的撰写始于国庆期间，当中由于工作过于繁忙而不断终止撰写，最近在设计另一个电商平台时再次萌发了完善此文并且发布此文的想法，期望自己的绵薄之力能够给予各位同行一些火花，共同推进国内的大型在线交易系统的研发工作，本文更多地站在软件工程角度来看待整个问题，有关后续的技术问题研究，将在另外的博文中予以探讨。**

   一年一度的国庆大假刚落下帷幕，由于这次长假是历史上最长的一次，因此出行问题备受关注，而铁路出行作为最主要的出行方式更是大家讨论的热点，老生常谈的购票难问题又被提起。这几天我在网站上也看到很多关于12306.cn的讨论，很多网友都发表了自己对于铁道部购票网站的不满，更有很多同行讨论了关于12306网站的设计问题，期待能够贡献自己的绵薄之力，我仔细拜读了其中至少10篇文章，很多同行多是站在技术的角度来考虑，其中不乏很多有创意的想法，纯粹的技术设计能解决一些问题，不过似乎不能够全面地解决这个庞大的、堪称瞬间流量“世界第一”的实时交易网站，目前12306的问题与其说是一个技术问题，还不如说它是个软件工程问题，道理非常简单，请看如下的新闻报导：

        [[[回望12306网站在2011年12月底以来的表现，铁道部高层也直呼想不到。    
    铁道部副部长胡亚东介绍，今年第一次在全国铁路实行网络电话订票，截至1月8日已经达到每天200万张，12306网站的注册用户已超过1000万人。1月1日至7日，“12306”网站日均点击次数已经超过了10亿次，专家认为瞬间点击可能达到了“世界第一”。    
      高度的关注、巨大的访问量，导致12306网站频繁出现系统崩溃、无法登陆、无法支付等情况。    
      “像春运这样庞大的需求量，难道铁道部没有预想到并有所准备？”隆梅资本管理有限公司副总经理马宏兴对此困惑不解。    
    在探究12306网站问题的深层原因以及解决之道时，各家看法不同，“12306网站的问题最终还是系统架构的问题。因为用户有大量的动态、交互式访问，所有的请求都会发送到12306网站的服务器端，同时在线并发用户数量太多，导致网站无力承载，造成拥堵。”华南师范大学计算机学院副院长单志龙认为。    
        又有说法认为，如果给12306网站增加服务器和带宽，也能够缓解拥堵的症状。这一观点铁道部内部颇为认同。    
    “得承认，我们对访问量估计得不足。”铁道部信息技术中心一位中层向记者透露，12306网站曾在2011年春运期间试运行，高峰时段访问量约在1亿点击量，因此，信息中心估计2012年春运期间的访问量约在3亿至4亿。    
    但是，结果却大大出乎人们的预料。“12306”网站在1月9日的日点击量达到14亿次，是原来料想峰值的5倍之多。“崩溃”在所难免！]]]  
  
笔者连日来也萌发了一个想法，假如让我来设计12306网站，我作为总架构师，该当如何考虑呢？自己虽然经历过众多的大项目的全生命周期跟踪管理，对于软件工程应该是有一定的研究，但像如此巨型项目，应该如何来设计、管控与实施？ 确实也颇伤脑筋，下面就笔者根据自己多年根植于IT研发的经验，特别是近年来对于巨型网站譬如国内的淘宝、京东等，国外的Facebook、Google等的跟踪研究经验谈谈自己的看法。  
  
1. 需求分析阶段  
需求分析是至关重要的，对于12306而言，需求分析的重点应该需要得出如下方面的关键数据才算需求分析基本结束：  
终端用户方面的：  
访问用户数量、总体注册用户数量、平时访问用户的峰值、平时访问用户的谷值、大假期访问用户的峰值、大假期访问用户的谷值、小假期访问用户的峰值、小假期访问用户的谷值；  
用户的地域分布性、用户可能介入的设备、用户接入的网络状况统计分析；  
后台服务方面的：  
关键购票流程业务分析：  
购票的基本流程分析、一次购票的TPS数量分析、一次购票的用户流量分析、一次购票的用户静态流量分析、一次购票的用户动态流量分析；  
这其中又分为初次购票与再次购票两种情况，流程稍微有些不同；  
系统提供的其他服务统计分析；  
前面所说的的大假在国内目前只有2个即春节与国庆，小假较多譬如清明、端午、中秋等。  
对于用户访问用户数、流量、网络、后台的TPS数量能够建立一个数学预测模型那就非常的清晰了，对于后续的设计指导意义至关重要；  
  
对于如此大的网站在安全性方面的需求需要做重点调研，另外由于是实时交易网站，还需要考虑金融安全问题，安全方面还得从如下两个方面来全面考虑：  
   内部安全，主要关注资金以及交易的安全，特别是防止内部人员尤其是系统管理员；  
   外部安全，主要关注如何确保拒绝外部恶意入侵与攻击成为一个核心，特别是类似DDOS之类的攻击；  
对照笔者的论述以及前面的新闻内容，不难发现，12306的设计组非常明显没有充分深入地进行需求调研与数据模型分析，特别是预案设计，因此笔者才敢说这更是个工程问题，而不完全是个纯粹的技术问题。  
  
2. 系统原型设计阶段  
国内很多的软件公司不太重视原型设计，这点对于小软件开发无关紧要，可是一旦到了大型软件的开发之时，不重视原型设计会失败得很惨，笔者曾经在后期救过一个ITSM管理系统的开发，究其原因，其中很重要的一条是对于原型设计不太重视，特别是在应用了大量的新技术而未进行原型设计是一件风险极大的事情，在笔者亲身带的几十个项目中有一部分就是这种情况，其中几个项目的痛苦经历（通宵达昼的加班长达1-2个月）更是令我终生难忘；  
  
根据前面的需求方面的分析讨论，不难发现12306的原型设计中需要解决的问题如下：  
前端Web服务器基于巨量访问的（秒均访问千万级别）可伸缩性模式验证：  
   这块可供选择的技术包括如下一些：基于CDN的Internet缓存加速技术、基于Apache Server+JBoss(Weblogic)的组合服务不同的、基于不同接口的调用原型研究、请求排队技术等的原型研究；  
  
后端数据库读写基于火车票应用的可扩展模式；  
   大家都知道，借鉴Facebook的巨量数据处理经验，必须基于现代数据库的分区技术、分布式处理技术并且通过后续的查询结果集成技术来实现巨量交易数据的可扩展性，基于巨量数据应用的可扩展模式通常有如下模式：  
  水平扩展技术，通常是基于分区技术来实施，维度是分区技术进行选择时的关键，针对于火车票的交易数据而言，时间维度是最好的选择，具体执行而言，可以将每天的车票交易信息放到某一分区上来执行，这样对于后续的数据维护（存储备份等）都将带来极大的方便；  
      垂直扩展技术，通常是基于地域等实现的分布式扩展，针对于火车票的交易数据而言，如何将不同地域的车票、车次信息划归为不同的数据库服务器来执行，确保在数据交易上的广域可并行性；  
  
对于12306系统，笔者建议最好能够按照列车始发与到达站的地域性进行垂直切割，而交易数据按照时间来进行横向水平扩展形成不同的子数据库，同时通过中间的缓存服务器、索引服务器、集成服务器将不同的数据进行地整合过来，提供给前端的应用服务器，因此从系统架构上来看这个结构图形如下：  
  
  
  
  
3. 原型验证阶段  
针对系统的原型设计，需要针对相应的子系统来验证原型的技术稳定性与成熟度，具体而言分别分为如下几段：  
针对前端访问的巨量级别的HTTP负载均衡子系统验证，特别是验证针对单个数据农场的服务响应能力，需要验证单个服务器的HTTP极限响应数量、动态扩展机制、网络带宽占用情况等；  
针对中间访问的缓存子系统、索引子系统、集成子系统，需要验证依照前端的用户请求如何将连接到后端不同的数据库上进行相应的数据分布化集成服务；  
针对后端访问的数据库垂直、水平切割技术，基于地域的垂直切割与基于时间的水平切割技术并且结合存储方面的分布式来扩充数据库系统的事务能力；  
  
4. 系统正式设计阶段  
正式设计整个系统时要考虑的设计方面还是挺多的，分别列出如下。  
功能性设计：  
前端Web服务器设计，可以按照Apache来负责静态页面响应处理、JBOSS/WEBLOGIC负责动态页面响应处理来进行设计，同时可以考虑将整个资源放到内存里面而使得Apache服务器对于静态资源的调用彻底离开磁盘I/O的制限，从而最大程度上提升前端Web服务器响应能力，这点笔者在一个游戏网站的后端服务器上曾经使用过，整体的web server响应能力提升了好几倍；  
具体的业务设计需要依照个业务的流程来拆解实施，此设计的关键是在于将前段的业务如何能够跟后端的高度扩展的分布数据库能够集成对应起来；  
  
后端数据库处理系统可以考虑按照如下的模式来予以完善设计：   将前段系统的运算分解与将各服务器进行（结果集成子系统）、使用成熟的反向搜索引擎系统作为前端的车站查询系统（搜索引擎子系统）、中间基于车次的具体查询可以使用缓存系统来实现（缓存子系统）、交易数据将写入到RDBMS之中（可水平、垂直扩展的事务处理子系统）； 这样设计的好处主要是将各种交易以及访问能够最大程度上的并行化，实现分布式集成化处理，从而最大程度上提升系统的整体处理能力；  
  
存储子系统的设计：  
主要是如何在RDMMS之间能够最大化各数据库的I/O处理能力同时提供不同地域（数据农场）的数据同步服务支持，同时对于从网络条件来看，建议将不同数据中心互联的光纤与用户请求的光纤分开了确保后端的数据同步响应不受前端的密集巨量请求服务之影响。  
  
安全性设计：  
前端的安全性设计主要包括防止诸如拒绝访问攻击(DDOS)、脚本注入攻击、用户信息安全、系统入侵等，后端的安全性设计主要是要考虑账户信息、交易的安全等；  
通常来说，前端可以考虑基于防火墙以及各种访问监测技术来做统计分析监控各种异常情况，而后端主要是基于数据加密、交易通道安全、数据库防篡改设计等来实施。  
  
冗余设计：  
包括依照地域、用户群建立不同的数据中心的设计、基于DNS区域解析规划设计、基于各服务器角色进行的冗余设计（WEB服务器、搜索服务器、缓存服务器、集成服务器、数据库服务器、存储服务器等）所进行的数据的冗余设计，譬如常见的集群技术、热备以及热切技术等均可考虑在此应用；  
其实笔者使用了中国国内的服务器以及美国的服务器分别解析了12306.cn域名地址，显示了两个不同的地址，这已说明12306设计组已经考虑了这些内容：  
从美国PING将会发现：  
ping www.12306.cn  
PING 08911.xdwscache.glb0.lxdns.com (183.60.136.64) 56(84) bytes of data.  
64 bytes from 183.60.136.64: icmp\_seq=1 ttl=49 time=171 ms  
64 bytes from 183.60.136.64: icmp\_seq=2 ttl=49 time=171 ms  
  
从上海PING将会发现：  
ping www.12306.cn  
Pinging 08911.xdwscache.glb0.lxdns.com [211.144.81.22] with 32 bytes of data:  
Reply from 211.144.81.22: bytes=32 time=11ms TTL=59  
Reply from 211.144.81.22: bytes=32 time=10ms TTL=59  
  
部署与维护设计：  
部署设计的关键是确保各种角色的服务器如何能够实现快速复制扩展，在紧急需要时能够快速部署实施，同时对于服务器集群在进行升级时如何快速批量地更新整个执行包；  
维护设计的关键考虑因素是系统运行的监控与报警系统设计，监控的指标就前端包括整个系统的访问数量、单个客户端的访问频率、访问的URL分布等，后端需要监控的是各服务器的使用量、负载量同时以此进行上报到监控服务器来决定相应的服务器是否需要进行动态扩展；  
  
5. 开发实施阶段  
针对于此巨型交易系统的开发，在开发阶段的关键因素是代码品质控制，建议采用软件工程成熟的Coding->Review->UT控制技术（基于TDD的开发模式）来完成个模块以及集成子系统的品质管控，在此特别提醒的是对于Review、UT段的密度需要做硬性的规定，流程裁剪时建议采用高密开发模式确保此阶段的品质管控目标，在此不得不多说一句的是很多时候很多的公司、开发人员总是错误地认为这些流程过于复杂，浪费时间，其实从各种实际的项目实施经历来看，此处多花了2-3倍的开发时间可以避免后续5-10甚至数十倍调试时间(Bug fix)的投入，整体上来算还是非常值得的，对于大型项目尤其如此，笔者在很多大型项目的集成阶段碰到集成不畅、无法按时完成集成任务时，开发、测试人员都是通宵达昼地加班，这其中很多的工作都是无用功，何不在开发实施阶段将系统的测试密度进行加大呢？构建质量上更好的组件（部件、模块），这样到集成的时候就不手忙脚乱了。  
  
由于此系统的规模非常大，因此在详细设计、开发时，应该考虑到各子系统必须设计得非常易于测试，特别是易于进行单体测试与自动化测试，这点可以大幅降低项目的执行成本;  
  
6. 测试阶段  
对此巨型系统的构建完毕后需要经历子系统集成测试、系统总集成测试两个阶段，并且需要按照如下不同的测试种类来进行逐个测试确认：  
 功能测试、性能测试、安全测试、破坏性测试、持久运行能力测试等  
功能测试需要针对多服务器角色集成后按照设计所提供的功能清单依据测试密度以及测试的分布性来确定不同方面的测试case，依据此来分步执行此功能测试，并且依照测试执行的结果来分析各个模块的缺陷密度以及缺陷的消化曲线，以此指导后期的开发设计工作，并藉此可尽快完成功能方面的验收确认；  
  
性能测试针对这个案列相对比较复杂，由于服务器角色很多，其基本的思路还是分析确定前端各业务的使用比例，编写开发自动化测试脚本来模拟用户的操作行为，并且在测试环境下针对特定的服务器数量的集群发起服务请求，同时完整地检测每个角色服务器的CPU/Memory/Disk I|O/Network方面的性能参数值，通过不断增减客户端请求数（伴随客户端机器的增加、譬如初期使用100台，逐步增加到200台测试机）来观察服务器的性能反应，分析出拐点，同时需要测试出系统的极限吞吐量(RPS,TPS等值)，并且测试出增加不同的服务器角色对于这些吞吐量的影响，同时在此阶段需要开发人员的协助来针对不同的角色根据测试结果进行性能调优工作；  
  
安全测试，需要针对以下不同的场景进行模拟攻击测试：  
外部安全而言： 包括脚本注入攻击、端口扫描、拒绝服务攻击、主机入侵等安全测试都需要逐个进行验证，确保系统的外部安全；  
内部安全而言： 包括各服务器的数据的安全、密码安全等进行测试；  
  
破坏性测试，需要就各个角色服务器所对应的物理服务器进行不同部位失效、移除等试验，看看整个服务器集群的监控、维护服务以及服务输出吞吐的影响，并且完善后续的各种系统对应的维护流程；  
  
持久运行能力测试需要，待系统达到Beta测试版本要求之后，需要针对待部署的系统进行一定负载量持续运行至少2周以上，并且观测系统的稳定性方面的反应，同时就数据库的增长、日志的增长等各种情况进行综合评估并且需要结合后续的部署维护做适当完善性调整；  
  
7. 部署阶段

对于如此巨型系统的部署调试工作，也是个不大不小的工程，那么如何确保部署调试工作进展的顺利，根据笔者多年维护广域网系统的经验，有如下几点需要注意：

 -> 如何管理好不同的服务器之间的发行版本问题，确保发布不会乱，特别是小版本的升级上部署不会乱是个关键，否则对于数以千计的服务器那绝对是个灾难；

 ->如何能够快速解决单次发布的升级与回退问题同样是很重要，这当中笔者的经验是使用一个经过验证的发布部署流程以及校验流程，整个过程最后放在一个发布脚本中，对于具体的升级与回退，维护人员只需要执行一次命令即可完成，并且对于每个版本发布完毕后的校验工作尤其重要，必须在脚本中予以体现；

 ->最后一点对于部署而言，任何部署都必须要有回滚方案配套，确保任何时间点上系统皆可用；

8. 运行维护阶段  
在运行维护阶段有很多细节需要配套考虑：  
  ->系统整体的实时状态监控，包括各种角色的应用主机的监控、网络设备的监控、用户访问流量的监控、服务可用性监控、安全监控等；  
  ->系统巨量的交易数据转储、交易日志的转储问题， 这个问题的解决需要结合前面在设计时的 数据库垂直（按时间维度）扩展以及日志按照天的维度来进行维护的方式进行有计划地转储到不同时间段的历史库中，并且通过数据的清洗、整理将部分重要数据转入到OLAP/OLTP系统中，为后续系统基于实时交易数据的联机分析改进业务提供帮助；  
  
9. 优化阶段

  ->业务流程优化：这块最主要的问题是将购票这个核心业务按照计算机易于执行计算的模型来进行不断地优化，同时体验，譬如异步事务提交优化、排队技术等皆可以使用在用户的模型上，同时可以向用户提供预订业务等来降低某个时段的网站流量压力，使得流量的整体峰值、谷值差距缩小很多；

  ->架构设计优化：在数据库的分区（垂直、水平分区）方面不断地进行优化，同时对于单数据库，就事务处理能力方面进行优化存储方面的操作，譬如使用固态硬盘替代传统的SCSI盘等，并行虚拟写技术等大幅提升磁盘I/O操作能力、或者采用内存数据库技术来管理好读写分离（读自内存、写到内存与磁盘同时），这样可以在数据库层面上获得性能的大幅提升；如果能够配合系统压力测试模型来获取其性能衰减曲线图针对性地优化，其效果将更加明显；

 在中间层服务器上，如何将不同的车次分布在大小几百个数据库上，并且使用反向搜索引擎技术来连接前端的访问请求到后端的数据库服务器之间的映射与集成（这点可以参考MapReduce并行算法）；

在应用层服务器上，如何将静态内容与动态内容予以剥离，并且无需保存的内容（只读）内容全放在内存中，如何平衡CDN加速与后端Web server的服务架构也是不断提升单个农场集群服务能力的关键与核心；

  ->性能优化： 这些优化是非常之多的，譬如针对Web server的socket连接池优化、日志优化，针对中间集成服务器的并行任务分解与结果集成优化，基于后端数据库服务器的数据计算模型、访问方式、冗余与数据同步、传递优化、数据分区优化等；

  ->其他调优：其他方面的调优譬如基于监控的优化、基于大并发量的快速响应与扩展优化等等；

写到此，感觉上似乎已经设计完毕了，不过突然回头一望，还真发现内容非常之多，可能有实战经验的朋友肯定会提到此系统属于“过度设计”了，不过笔者在此要说的是，确实这个最后的提醒是非常有必要的，任何系统的设计特别是如此复杂的系统的设计只能是需要通过时间来循序渐进，先原型后实际系统并且遵循几轮研发调优（含架构设计调优），最后逐渐演变到一个真实成熟的系统，任何超前的设计最终都被证明是无用功而遭废弃；  
  
很多的网友对于12306的网站的技术议论纷纭，并且提出了众多的创造性想法，笔者在提出自己的想法后还有几点补充想法供大家参考：  
1. 其实此网站的需求不是一个纯粹的技术问题，试想一下，真的按照国庆、春节的高峰访问需求来设计、部署整个系统的时候，在平时的时候大部分机器都是浪费的，如何平衡峰值与谷值时矛盾是个技术问题，可背后掩盖着的是全中国13亿多人口在短短的10-15天内完成众多的人口迁徙，如此巨大的工程放在任何一个国家都是难于解决的问题，如何从需求层面来化解这个短期内的巨量人口流动问题才是此问题的命根； 正所谓 “ 解决火车票购买过程容易，而解决人人能买到火车票难！！！”  
2. 此问题的求解永远是个迭代过程，需要看到的是很难有人能够在短期内设计并且实现这么一个巨量系统，这个工程问题的求解是拆解为不同的子模块来分步实现，并且逐步迭代优化求解的过程；

3. 此问题的解决必须是个系统工程，牵涉到资金、技术、人才、时间 这4个基本要素，很多的同行过于看重人才与技术，而忽略了资金等前提性因素，试想如果只有3000万预算让你来设计实现这个巨型交易系统，你能解决么？ 不说别的，光性能模拟测试就需要调动数以万计的前端机器来在全国范围内不同的典型区域发起海量请求，这些是个小资金能够解决得了的问题么？

**后记：**笔者深知，任何技术的讨论必然会引起一些争议，整个技术的发展必须要经历这样的纷争才能进步与提高，在此博文发布的几天内，备受各位同行的关注，笔者非常感谢，也发现了诸多不友好的言辞， 对于有类似架构经验、大项目经历的同行一看就明白，而无此经历的同行看起来相对比较而言没有感觉，笔者在此期望技术纷争的同行不要爆粗，大家共同的目的是为了推动技术交流，推动你我的技术进步，推动国家与民族的技术发展。

=====================评论=================================

写得很全面哦，分析得不错，我也想到了几点。  
1. 最关键的，是人为的设置了10天内车票可以订购这个门槛，就把压力，尤其是紧张时期的订票压力，都集中到了10天前的那几个小时，后面都已经订不到了；所以应该放开订购，至少放到3个月到半年，这样压力可以大大缓解；  
2. 数据库中，分最核心的服务器和外围服务器，最核心的部分是提交票用的，利用数据库同步技术把票的情况备份到外围服务器；所有的查询操作在外围服务器；因为绝大部分都是查询请求，而外围服务器可以部署很多，基本只读，做分布式完全没有压力；  
3. 核心服务器也可以再根据余票情况做1级、2级服务，当余票较多时，2级服务器就可以先从1级服务器批量获取一批票，直接对外响应发票，再定时分布提交到1级服务器；  
4. 对应用服务器和web server，这个关键还是做过负荷和负载均衡，这些现成的工具都很多，也不是很贵。其实每个机器的TPMC和吞吐量都可以计算出来，将消息队列大小计算后，给个合理值，外围的请求必须过负荷掉，防止对正常业务造成冲击；每个系统都是有设计能力的，在能力外的不接入大家也是能接受的。另外，将查询服务器和订票服务器分开后，登陆之类的基本上就不会有什么瓶颈了，因为查询服务器可以平滑扩展的，订票的资源有限就决定了必定需要过负荷的。

Re: [jackxinxu2100](http://blog.csdn.net/jackxinxu2100) 2012-12-04 09:36发表 [[回复]](http://blog.csdn.net/jackxinxu2100/article/details/8140755#reply) [[引用]](http://blog.csdn.net/jackxinxu2100/article/details/8140755#quote) [[举报]](http://blog.csdn.net/jackxinxu2100/article/details/8140755#report)

[http://avatar.csdn.net/9/2/F/3_jackxinxu2100.jpg](http://blog.csdn.net/jackxinxu2100)

回复lxpgao1：嗯，你使用了定量计算的方法来匹配计算能力，值得肯定的，在概要设计时确实要用到。