关于淘宝的，详细可以参考如下

一、个人网站

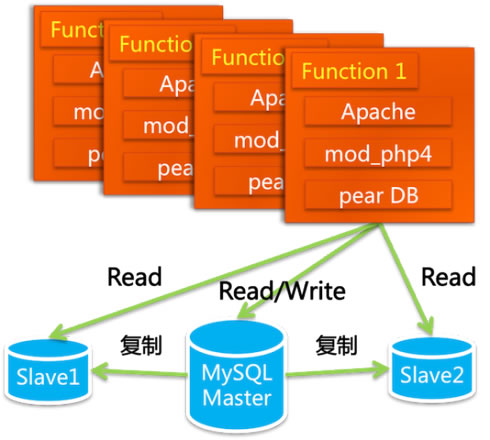
　　2003 年 4 月 7 日，马云，在杭州，成立了一个神秘的组织。他叫来十位员工，要他们签了一份协议，这份协议要求他们立刻离开阿里巴巴，去做一个神秘的项目。这个项目要求绝对保密，老马戏称“连说梦话被老婆听到都不行，谁要是透漏出去，我将追杀到天涯海角”。这份协议是英文版的，匆忙之间，大多数人根本来不及看懂，但出于对老马的信任，都卷起铺盖离开了阿里巴巴。

　　他们去了一个神秘的据点 —— 湖畔花园小区的一套未装修的房子里，房子的主人是马云。这伙人刚进去的时候，马云给他们布置了一个任务，就是在最短的时间内做出一个个人对个人（C2C）的商品交易的网站。现在出一个问题考考读者，看你适不适合做淘宝的创业团队。亲，要是让你来做，你怎么做？

　　在说出这个答案之前，容我先卖个关子，介绍一下这个创业团队的成员：三个开发工程师（虚竹、三丰、多隆）、一个UED（二当家）、三个运营（小宝、阿珂、破天）、一个经理（财神）、还有就是马云和他的秘书。当时对整个项目组来说压力最大的就是时间，怎么在最短的时间内把一个从来就没有的网站从零开始建立起来？了解淘宝历史的人知道淘宝是在 2003 年 5 月 10 日上线的，这之间只有一个月。要是你在这个团队里，你怎么做？我们的答案就是：买一个来。

　　买一个网站显然比做一个网站要省事一些，但是他们的梦想可不是做一个小网站而已，要做大，就不是随便买个就行的，要有比较低的维护成本，要能够方便的扩展和二次开发。那接下来就是第二个问题：买一个什么样的网站？答案是：轻量一点的，简单一点的，于是买了这样一个[**架构**](http://lib.csdn.net/base/architecture)的网站：LAMP([**Linux**](http://lib.csdn.net/base/linux)+Apache+[**MySQL**](http://lib.csdn.net/base/mysql)+[**PHP**](http://lib.csdn.net/base/php))。这个直到现在还是一个很常用的网站架构模型。这种架构的优点是：无需编译，发布快速，PHP功能强大，能做从页面渲染到数据访问所有的事情，而且用到的技术都是开源的，免费。

　　当时我们是从一个美国人那里买来的一个网站系统，这个系统的名字叫做 PHPAuction（他们的官方网站 http://www.phpauction[**.NET**](http://lib.csdn.net/base/dotnet)，这个名字很直白，一眼就看出来这个系统是用什么语言做的、是干什么用的），PHPAuction有好几个版本，我们买的是最高版的，功能比较多，而且最重要的是对方提供了源代码。最高版比较贵，花了我们 2000 美金（貌似现在降价了，只要 946 美元）。买来之后不是直接就能用的，需要很多本地化的修改，例如页面模板改的漂亮一点，页头页脚加上自己的站点简介等，其中最有技术含量的是对[**数据库**](http://lib.csdn.net/base/mysql)进行了一个修改。原来是从一个数据库进行所有的读写操作，拿过来之后多隆把它给拆分成一个主库、两个从库，读写分离。这么做的好处有几点：存储容量增加了，有了备份，使得安全性增加了，读写分离使得读写效率提升了。这样整个系统的架构就如下图所示：

[](http://static.oschina.net/uploads/img/201411/13161405_fHq0.jpg)

　　其中 Pear DB 是一个 PHP 模块，负责数据访问层。另外也用开源的论坛系统 PHPBB（http://www.phpbbchina.com ）搭建了一个小的论坛社区，虚竹负责机器采购、配置、架设等，三丰和多隆负责编码，他们把交易系统和论坛系统的用户信息打通，给运营人员开发出后台管理（admin系统）的功能，把交易类型从只有拍卖这一种增加为拍卖、一口价、求购商品、海报商品（意思是还没推出的商品，先挂个海报出来）这四种。（PHPAuction 只有拍卖的交易，Auction 即拍卖的意思。@\_行癫在微博中提到：今天 eBay 所有交易中拍卖交易仍然占了 40%，而在中国，此种模式在淘宝几乎从一开始就未能占据优势，如今在主流的交易中几乎可以忽略不计。背后的原因一直令人费解。我大致可以给出其中一种解释，eBay 基本在发达国家展开业务，制造业外包后，电子商务的基本群体大多只能表现为零散的个体间交易。）

　　在经历了另外一些有趣的事情之后（这些有趣的事情包括“淘宝”这个名字的由来，员工花名的由来等等，由于本书主要描述技术方面的故事，对这些有兴趣的可以去网上找），网站开始上线运行了。

[](http://static.oschina.net/uploads/img/201411/13161405_veK0.jpg)

　　在接下来的大半年时间里，这个网站迅速显示出了它的生机。这里有必要提一下当时的市场环境，非典（SARS）的肆虐使得大家都不敢出门，尤其是去商场之类人多的地方。另外在神州大地上最早出现的 C2C 网站易趣也正忙的不亦乐乎，2002 年 3 月，eBay 以 3000 万美元收购了易趣公司 33% 的股份，2003 年 6 月以 1.5 亿美元收购了易趣公司剩余 67% 的股份。当时淘宝网允许买卖双方留下联系方式，允许同城交易，整个操作过程简单轻松。而 eBay 为了收取交易佣金，是禁止这么做的，这必然增加了交易过程的难度。而且 eBay 为了全球统一，把易趣原来的系统替换成了美国 eBay 的系统，用户体验一下子全变了，操作起来非常麻烦，这等于是把积累的用户拱手送给了淘宝。为了不引起 eBay 的注意，淘宝网在 2003 年里一直声称自己是一个“个人网站”。由于这个创业团队强大的市场开拓和运营能力，淘宝网发展的非常迅猛，2003 年底就吸引了注册用户XXX，最高每日 31 万PV，从 5 月到年底成交额 4000 万。这没有引起 eBay 的注意，却引起了阿里巴巴内部很多员工的注意，他们觉得这个网站以后会成为阿里巴巴强劲的对手。甚至有人在内网发帖，忠告管理层要警惕这个刚刚起步的网站，但管理层似乎无动于衷。（这个团队的保密工作做的真好）

　　在市场和运营的后方，淘宝网的技术团队也在快速的做着系统的改进和创新。这里还有个有趣的故事，eBay 和易趣早期都有员工在论坛上响应用户的需求，eBay 的论坛用粉红色背景来区分员工的发言，易趣的员工在论坛上昵称都选各种豆豆，例如黄豆豆、蚕豆豆等。淘宝在讨论运营策略的时候提到这个问题，要求所有的员工都去论坛上回答用户的问题。最早回答问题的任务落在小宝头上，那我们用什么名字好呢？“淘淘”？“宝宝”？小宝都不满意，太女性化了。讨论了很久之后，小宝灵光乍现，干脆取个名字叫“小宝”吧，小宝带七个老婆来开店，迎接各位客官，很有故事性。于是很多武侠小说中的人物开始在论坛中行侠仗义，这些昵称下面标志着“淘宝店小二”，他们回答着各种各样的问题，快速响应着用户的各种需求。如果是技术上能解决的，几个人商量一下，马上就开发、[**测试**](http://lib.csdn.net/base/softwaretest)、发布上线。反过来对比一下，易趣被 eBay 收购之后，系统更换成了全球通用的版本，响应用户的一个需求需要层层审批，反应速度自然慢了下来。

　　当时淘宝第一个版本的系统里面已经包含了商品发布、管理、搜索、商品详情、出价购买、评价投诉、我的淘宝这些功能（现在主流程中也是这些模块。在 2003 年 10 月增加了一个功能节点：“安全交易”，这个是支付宝的雏形）。随着用户需求和流量的不断增长，系统上面做了很多的日常改进，服务器由最初的一台变成了三台，一台负责发送 email、一台负责运行数据库、一台负责运行 Web App。过一段时间之后，商品搜索的功能占用数据库资源太大了（用like搜索的，很慢），又从阿里巴巴中文站搬过来他们的[**搜索引擎**](http://lib.csdn.net/base/searchengine) iSearch，起初 iSearch 索引的文件放在硬盘上，随着数据量的增长，又采购了 NetApp 服务器放置 iSearch。

　　如此快节奏的工作，其实大家都累得不行，有人就提议大家随时随地的锻炼身体，可是外面 SARS 横行，在一个一百多方的房子里，怎么锻炼呢？高挑美女阿珂提议大家练习提臀操，这个建议遭到男士的一致反对，后来虚竹就教大家练习倒立，这个大家都能接受。于是这个倒立的传统一直延续至今，和花名文化、武侠文化一并传承了下来。

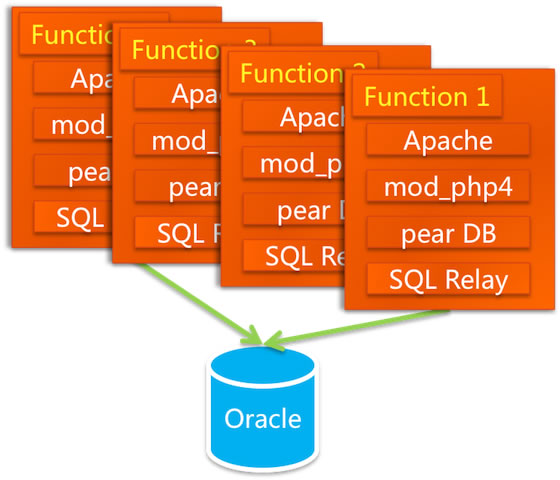
　　随着访问量和数据量的飞速上涨，问题很快就出来了，第一个问题出现在数据库上。MySQL 当时是第 4 版的，我们用的是默认的存储引擎 MyISAM，这种类型读数据的时候会把表锁住（我们知道 [**Oracle**](http://lib.csdn.net/base/oracle) 在写数据的时候会有行锁，读数据的时候是没有的），尤其是主库往从库上面写数据的时候，会对主库产生大量的读操作，使得主库性能急剧下降。这样在高访问量的时候，数据库撑不住了。另外，当年的 MySQL 不比如今的 MySQL，在数据的容量和安全性方面也有很多先天的不足（和 Oracle 相比）。

　　二、Oracle/支付宝/旺旺

　　淘宝网作为个人网站发展的时间其实并不长，由于它太引人注目了，马云在 2003 年 7 月就宣布了这个是阿里巴巴旗下的网站，随后在市场上展开了很成功的运作。最著名的就是利用中小网站来做广告，突围 eBay 在门户网站上对淘宝的广告封锁。上网比较早的人应该还记得那些在右下角的弹窗和网站腰封上一闪一闪的广告。市场部那位到处花钱买广告的家伙，太能花钱了，一出手就是几百万，他被我们称为“大少爷”。

　　“大少爷”们做的广告，带来的就是迅速上涨的流量和交易量。在 2003 年底，MySQL 已经撑不住了，技术的替代方案非常简单，就是换成 Oracle。换 Oracle 的原因除了它容量大、稳定、安全、性能高之外，还有人才方面的原因。在 2003 年的时候，阿里巴巴已经有一支很强大的 DBA 团队了，有冯春培、汪海（七公）这样的人物，后来还有冯大辉（@fenng）、陈吉平（拖雷）。这样的人物牛到什么程度呢？Oracle 给全球的技术专家颁发一些头衔，其中最高级别的叫 ACE（就是扑克牌的“尖儿”，够大的吧），被授予这个头衔的人目前全球也只有 300 多名（名单在这里： <http://apex.oracle.com/pls/otn/f?p=19297:3> ），当年全球只有十几名。有如此强大的技术后盾，把 MySQL 换成 Oracle 是顺理成章的事情。

　　但更换数据库不是只换个库就可以的，访问方式，SQL 语法都要跟着变，最重要的一点是，Oracle 并发访问能力之所以如此强大，有一个关键性的设计 —— 连接池。但对于 PHP 语言来说它是放在 Apache 上的，每一个请求都会对数据库产生一个连接，它没有连接池这种功能（[**Java**](http://lib.csdn.net/base/javase) 语言有 Servlet 容器，可以存放连接池）。那如何是好呢？这帮人打探到 eBay 在 PHP 下面用了一个连接池的工具，是 BEA 卖给他们的。我们知道 BEA 的东西都很贵，我们买不起，于是多隆在网上寻寻觅觅，找到一个开源的连接池代理服务 SQLRelay（ [http://sourceforge.jp/projects/freshmeat\_sqlrelay](http://sourceforge.jp/projects/freshmeat_sqlrelay/) ），这个东西能够提供连接池的功能，多隆对它进行了一些功能改进之后就拿来用了。这样系统的架构就变成了如下的样子：

[](http://static.oschina.net/uploads/img/201411/13161406_GK8j.jpg)

　　数据一开始是放在本地的，DBA 们对 Oracle 做调优的工作，也对 SQL 进行调优。后来数据量变大了，本地存储不行了。买了 NAS（Network Attached Storage：网络附属存储），NetApp 的 NAS 存储作为了数据库的存储设备，加上 Oracle RAC（Real Application Clusters，实时应用集群）来实现负载均衡。七公说这实际上是走了一段弯路，NAS 的 NFS（Network File System）协议传输的延迟很严重，但那时侯不懂。后来采购了 Dell 和 EMC 合作的 SAN 低端存储，性能一下子提升了 10 几倍，这才比较稳定了。再往后来数据量更大了，存储的节点一拆二、二拆四，RAC 又出问题了。这才踏上了购买小型机的道路。在那段不稳定的时间里，七公曾经在机房住了 5 天 5 夜。

　　替换完数据库，时间到了 2004 年春天，俗话说“春宵一刻值千金”，但这些人的春宵却不太好过了。他们在把数据的连接放在 SQLRelay 之后就噩梦不断，这个代理服务经常会死锁，如同之前的 MySQL 死锁一样。虽然多隆做了很多修改，但当时那个版本内部处理的逻辑不对，问题很多，唯一解决的办法就是“重启”它的服务。这在白天还好，连接上机房的服务器，把进程杀掉，然后开启就可以了，但是最痛苦的是它在晚上也要死掉，于是工程师们不得不 24 小时开着手机，一旦收到“ SQLRelay 进程挂起”的短信，就从春梦中醒来，打开电脑，连上机房，重启服务。后来干脆每天睡觉之前先重启一下。做这事最多的据说是三丰，他现在是淘宝网的总裁。现在我们知道，任何牛B的人物，都有一段苦B的经历。

　　微博上有人说“好的架构是进化来的，不是设计来的”。的确如此，其实还可以再加上一句“好的功能也是进化来的，不是设计来的”。在架构的进化过程中，业务的进化也非常迅猛。最早的时候，买家打钱给卖家都是通过银行转账汇款，有些骗子收了钱却不发货，这是一个很严重的问题。然后这伙人研究了 PayPal 的支付方式，发现也不能解决问题。后来这几个聪明的脑袋又想到了“担保交易”这种第三方托管资金的办法。于是在 2003 年 10 月，淘宝网上面上线了一个功能，叫做“安全交易”，卖家选择支持这种功能的话，买家会把钱交给淘宝网，等他收到货之后，淘宝网再把钱给卖家。这就是现在的支付宝，在前两天（2012.2.21）年会上，支付宝公布 2011 年的交易笔数已经是 PayPal 的两倍。这个划时代的创新，其实就是在不断的思索过程中的一个灵光乍现。

　　当时开发“安全交易”功能的是茅十八和他的徒弟苗人凤（茅十八开发到一半去上海读 MBA 去了，苗人凤现在是支付宝的首席业务架构师），开发跟银行网关对接的功能的是多隆。当时多数银行的网站已经支持在线支付了，但多隆告诉我，他们的网关五花八门，用什么技术的都有，必须一家一家去接。而且他们不保证用户付钱了就一定扣款成功、不保证扣款成功了就一定通知淘宝、不保证通知淘宝了就一定能通知到、不保证通知到了就不重复通知。这害苦了苗人凤，他必须每天手工核对账单，对不齐的话就一定是有人的钱找不到地方了，少一分钱都睡不着觉。另外他为了测试这些功能，去杭州所有的银行都办理了一张银行卡。一堆银行卡摆在桌子上，不知道的人还以为这个家伙一定很有钱，其实里面都只是十块八块的。现在我们再一次知道，任何牛B的人物，都必须有一段苦B的经历。

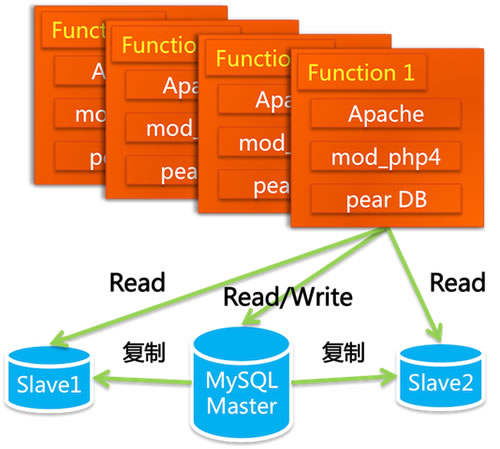
　　有人说淘宝打败易趣（eBay 中国）是靠免费，其实这只是原因之一。如果说和易趣过招第一招是免费的话，这让用户没有门槛就愿意来，那第二招就是“安全支付”，这让用户放心付款，不必担心被骗。在武侠小说中真正的高手飞花摘叶即可伤人，他们不会局限于一招两招，一旦出手，连绵不绝。而淘宝的第三招就是“旺旺”，让用户在线沟通。其实淘宝旺旺也不是自己生出来的，是从阿里巴巴的“贸易通”复制过来的。从 2004 年 3 月开始，“叮咚、叮咚”这个经典的声音就回荡在所有淘宝买家和卖家的耳边，“亲，包邮不？”，“亲，把零头去掉行不？”，这亲切的砍价声造就了后来的“淘宝体”。有人说中国人就是爱砍价，虽然笔者体会不到砍价成功后有多少成就感，但每次我去菜市场，看到大妈们砍价砍得天昏地暗，那满足的劲头堪比捡到了钱，我就深刻的理解了淘宝旺旺在交易过程中的价值。我猜 eBay 也体会不到砍价的乐趣，他们一直不允许买卖双方在线聊天，收购了 skype 之后也没有用到电子商务中去。

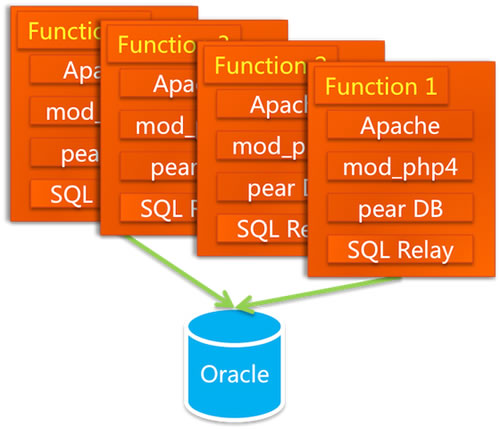
　　旺旺在推出来没多久，就惹了一个法律方面的麻烦。有个做雪饼的厂家找上门来，说我们侵权了，他们家的雪饼很好吃，牛奶也做得不错，我们都很喜欢。然后我们就在旺旺的前面加了两个字，叫做“淘宝旺旺”。在那个野蛮生长的阶段，其实很多产品都是想到什么就做什么，例如我们还搭建过一个聊天室，但似乎淘宝网不是一个闲聊的地方，这个聊天室门可罗雀，一段时间后就关闭掉了。

　　SQLRelay 的问题搞得三丰他们很难睡个囫囵觉，那一年开半年会的时候，公司特地给三丰颁了一个奖项，对他表示深切的安慰。但不能总这样啊，于是，2004 年的上半年开始，整个网站就开始了一个脱胎换骨的手术。

　　三、淘宝技术发展（Java时代：脱胎换骨）

　　我的师父黄裳@岳旭强曾经说过，“好的架构图充满美感”，一个架构好不好，从审美的角度就能看得出来。后来我看了很多系统的架构，发现这个言论基本成立。那么反观淘宝前面的两个版本的架构，你看哪个比较美？

[](http://static.oschina.net/uploads/img/201411/13161406_qhKT.jpg)

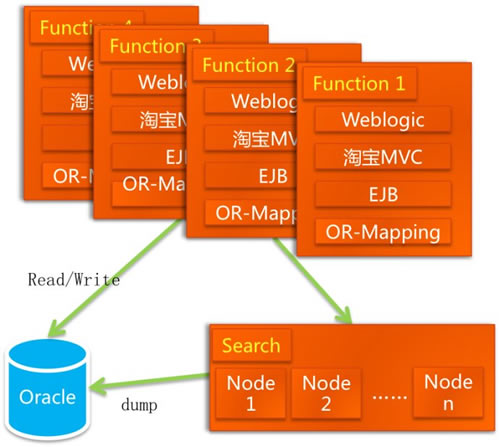
[](http://static.oschina.net/uploads/img/201411/13161406_WPI7.jpg)

　　显然第一个比较好看，后面那个显得头重脚轻，这也注定了它不是一个稳定的版本，只存活了不到半年的时间。2004 年初，SQL Relay 的问题解决不了，数据库必须要用 Oracle，那从哪里动刀？只有换开发语言了。换什么语言好呢？Java。Java 是当时最成熟的网站开发语言，它有比较良好的企业开发框架，被世界上主流的大规模网站普遍采用，另外有 Java 开发经验的人才也比较多，后续维护成本会比较低。

　　到 2004 年上半年，淘宝网已经运行了一年的时间，这一年积累了大量的用户，也快速的开发了很多功能，当时这个网站已经很庞大了，而且新的需求还在源源不断的过来。把一个庞大的网站的开发语言换掉，无异于脱胎换骨，在换的过程中还不能拖慢业务的发展，这无异于边换边跑，对时间和技术能力的要求都非常高。做这样的手术，需要请第一流的专家来主刀。现在再考一下读者，如果你在这个创业团队里面，请什么样的人来做这事？我们的答案是请 Sun 的人。没错，就是创造 Java 语言的那家公司，世界上没有比他们更懂 Java 的了。除此之外，还有一个不为人知的原因，……（此处和谐掉 200 字，完整版见 aliway）

　　这帮 Sun 的工程师的确很强大，在笔者 2004 年底来淘宝的时候，他们还在，有幸跟他们共事了几个月。现在摆在他们面前的问题是用什么办法把一个庞大的网站从 PHP 语言迁移到 Java？而且要求在迁移的过程中，不停止服务，原来系统的 bugfix 和功能改进不受影响。亲，你要是架构师，你怎么做？有人的答案是写一个翻译器，如同把中文翻译成英文一样，自动翻译。我只能说你这个想法太超前了，换个说法就是“too simple, sometimes naive”。当时没有，现在也没有人能做到。他们的大致方案是给业务分模块，一个模块一个模块的替换。如用户模块，老的 member.taobao.com 继续维护，不添加新功能，新的功能先在新的模块上开发，跟老的共用一个数据库，开发完毕之后放到不同的应用集群上，另开个域名 member1.taobao.com，同时替换老的功能，替换一个，把老的模块上的功能关闭一个，逐渐的把用户引导到 member1.taobao.com，等所有功能都替换完毕之后，关闭 member.taobao.com。后来很长时间里面都是在用 member1 这样奇怪的域名，两年后有另外一家互联网公司开始做电子商务了，我们发现他们的域名也叫 member1.xx.com、auction1.xx.com……

　　说了开发模式，再说说用到的 [**Java**](http://lib.csdn.net/base/java)MVC 框架，当时的 Struts 1.x 是用的比较多的框架，但是用过 WebWork 和 Struts 2 的同学可能知道，Struts 1.x 在多人协作方面有很多致命的弱点，由于没有一个轻量框架作为基础，因此很难扩展，这样架构师对于基础功能和全局功能的控制就很难做到。而阿里巴巴的 18 个创始人之中，有个架构师，在 Jakarta Turbine 的基础上，做了很多扩展，打造了一个阿里巴巴自己用的 MVC 框架 WebX （<http://www.openwebx.org/docs/Webx3_Guide_Book.html>），这个框架易于扩展，方便组件化开发，它的页面模板支持 JSP 和 Velocity 等、持久层支持 iBATIS 和 [**hibernate**](http://lib.csdn.net/base/javaee) 等、控制层可以用 EJB 和 [**spring**](http://lib.csdn.net/base/javaee)（Spring 是后来才有的）。项目组选择了这个强大的框架，这个框架如果当时开源了，也许就没有 WebWork 和 Struts 2 什么事了。另外，当时 Sun 在全世界大力推广他们的 EJB，虽然淘宝的架构师认为这个东东用不到，但他们还是极力坚持。在经历了很多次的技术讨论、争论和争吵之后，这个系统的架构就变成了下图的样子：

[](http://static.oschina.net/uploads/img/201411/13161406_MVQj.jpg)

　　Java 应用服务器是 Weblogic，MVC 框架是 WebX、控制层用了 EJB、持久层是 iBATIS，另外为了缓解数据库的压力，商品查询和店铺查询放在搜索引擎上面。这个架构图是不是好看了一点了，亲？

　　这帮 Sun 的工程师开发完淘宝的网站之后，又做了一个很牛的网站，叫“支付宝”。

　　其实在任何时候，开发语言本身都不是系统的瓶颈，业务带来的压力更多的是压到了数据和存储上。上面一篇也说到，MySQL 撑不住了之后换 Oracle，Oracle 的存储一开始在本机上，后来在 NAS 上，NAS 撑不住了用 EMC 的 SAN 存储，再然后 Oracle 的 RAC 撑不住了，数据的存储方面就不得不考虑使用小型机了。在 2004 年的夏天，DBA 七公、测试工程师郭芙和架构师行癫，踏上了去北京测试小型机的道路。他们带着小型机回来的时候，我们像欢迎领袖一样的欢迎他们，因为那个是我们最值钱的设备了，价格表上的数字吓死人。小型机买回来之后我们争相合影，然后 Oracle 就跑在了小型机上，存储方面从 EMC 低端 cx 存储到 Sun oem hds 高端存储，再到 EMC dmx 高端存储，一级一级的往上跳。

　　到现在为止，我们已经用上了 IBM 的小型机、Oracle 的数据库、EMC 的存储，这些东西都是很贵的，那些年可以说是花钱如流水啊。有人说过“钱能解决的问题，就不是问题”，但随着淘宝网的发展，在不久以后，钱已经解决不了我们的问题了。花钱买豪华的配置，也许能支持 1 亿 PV 的网站，但淘宝网的发展实在是太快了，到了 10 亿怎么办？到了百亿怎么办？在 N 年以后，我们不得不创造技术，解决这些只有世界顶尖的网站才会遇到的问题。后来我们在开源软件的基础上进行自主研发，一步一步的把 IOE（IBM 小型机、Oracle、EMC 存储）这几个“神器”都去掉了。这就如同在《西游记》里面，妖怪们拿到神仙的兵器会非常厉害，连猴子都能够打败，但最牛的神仙是不用这些神器的，他们挥一挥衣袖、翻一下手掌就威力无比。去 IOE 这一部分会在最后一个章节里面讲，这里先埋个千里伏笔。

　　欲知后事如何，且听下回分解。

　　四、淘宝技术发展（Java时代：坚若磐石）

　　已经有读者在迫不及待的问怎么去掉了 IOE，别急，在去掉 IOE 之前还有很长的路要走。行癫他们买回来小型机之后，我们用上了 Oracle，七公带着一帮 DBA 在优化 SQL 和存储，行癫带着几个架构师在研究数据库的扩展性。Oracle 本身是一个封闭的系统，用 Oracle 怎么做扩展？用现在一个时髦的说法就是做“分库分表”。

　　我们知道一台 Oracle 的处理能力是有上限的，它的连接池有数量限制，查询速度跟容量成反比。简单的说，在数据量上亿、查询量上亿的时候，就到它的极限了。要突破这种极限，最简单的方式就是多用几个 Oracle 数据库。但一个封闭的系统做扩展，不像分布式系统那样轻松。我们把用户的信息按照 ID 来放到两个数据库里面（DB1/DB2），把商品的信息跟着卖家放在两个对应的数据库里面，把商品类目等通用信息放在第三个库里面(DBcommon)。这么做的目的除了增加了数据库的容量之外，还有一个就是做容灾，万一一个数据库挂了，整个网站上还有一半的数据能操作。

　　数据库这么分了之后，应用程序有麻烦了，如果我是一个买家，买的商品有 DB1 的也有 DB2 的，要查看“我已买到的宝贝”的时候，应用程序怎么办？必须到两个数据库里面分别查询出来对应的商品。要按时间排序怎么办？两个库里面“我已买到的宝贝”全部查出来在应用程序里面做合并。还有分页怎么处理？关键字查询怎么处理？这些东西交给程序员来做的话会很悲催，于是行癫在淘宝的第一个架构上的作品就来解决了这个问题，他写了一个数据库路由的框架 DBRoute，这个框架在淘宝的 Oracle 时代一直在使用。后来随着业务的发展，这种分库的第二个目的 —— 容灾的效果就没有达到。像评价、投诉、举报、收藏、我的淘宝等很多地方，都必须同时连接 DB1 和 DB2，哪个库挂了都会导致整个网站挂掉。

　　上一篇说过，采用 EJB 其实是和 Sun 的工程师妥协的结果，在他们走了之后，EJB 也逐渐被冷落了下来。在 2005、2006年的时候，Spring 大放异彩，正好利用 Spring 的反射（IoC）模式替代了 EJB 的工厂模式，给整个系统精简了很多代码。

　　上一篇还说过，为了减少数据库的压力，提高搜索的效率，我们引入了搜索引擎。随着数据量的继续增长，到了 2005 年，商品数有 1663 万，PV 有 8931 万，注册会员有 1390 万，这给数据和存储带来的压力依然山大，数据量大，性能就慢。亲，还有什么办法能提升系统的性能？一定还有招数可以用，这就是缓存和 CDN（内容分发网络）。

　　你可以想象，九千万的访问量，有多少是在商品详情页面？访问这个页面的时候，数据全都是只读的（全部从数据库里面读出来，不写入数据库），如果把这些读操作从数据库里面移到内存里，数据库将会多么的感激涕零。在那个时候我们的架构师多隆大神，找到了一个基于 Berkeley DB 的开源的缓存系统，把很多不太变动的只读信息放了进去。其实最初这个缓存系统还比较弱，我们并没有把整个商品详情都放在里面，一开始把卖家的信息放里面，然后把商品属性放里面，商品详情这个字段太大，放进去受不了。说到商品详情，这个字段比较恐怖，有人统计过，淘宝商品详情打印出来平均有 5 米长，在系统里面其实放在哪里都不招人待见。笔者清楚的记得，我来淘宝之后担任项目经理做的第一个项目就是把商品详情从商品表里面给移出来。这个字段太大了，查询商品信息的时候很多都不需要查看详情，它跟商品的价格、运费这些放在一个表里面，拖慢了整个表的查询速度。在 2005 年的时候，我把商品详情放在数据库的另外一张表里面，再往后这个大字段被从数据库里面请了出来，这也让数据库再一次感激涕零。

　　到现在为止，整个商品详情的页面都在缓存里面了，眼尖的读者可能会发现现在的商品详情不全是“只读”的信息了，这个页面上有个信息叫“浏览量”，这个数字每刷新一次页面就要“写入”数据库一次，这种高频度实时更新的数据能用缓存吗？如果不用缓存，一天几十亿的写入，数据库会怎么样？一定会挂掉。那怎么办？亲……先不回答你（下图不是广告，让你看看浏览量这个数据在哪里）

[](http://static.oschina.net/uploads/img/201411/13161406_oCoO.jpg)

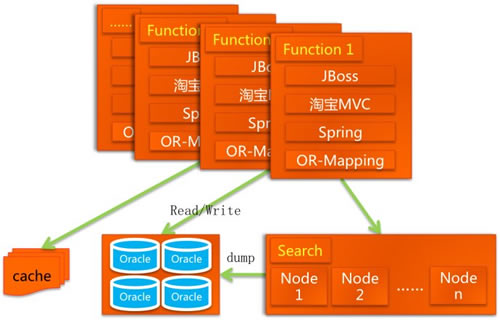
　　CDN 这个工作相对比较独立，跟别的系统一样，一开始我们也是采用的商用系统。后来随着流量的增加，商用的系统已经撑不住了，LVS 的创始人章文嵩博士带人搭建了淘宝自己的 CDN 网络。在本文的引言中我说过淘宝的 CDN 系统支撑了 800Gbps 以上的流量，作为对比我们可以看一下国内专业做 CDN 的上市公司 ChinaCache 的介绍 —— “ChinaCache……是中国第一的专业 CDN 服务提供商，向客户提供全方位网络内容快速分布解决方案。作为首家获信产部许可的 CDN 服务提供商，目前 ChinaCache 在全国 50 多个大中城市拥有近 300 个节点，全网处理能力超过 500Gbps，其 CDN 网络覆盖中国电信、中国网通、中国移动、中国联通、中国铁通和中国教育科研网等各大运营商。” —— 这样你可以看得出淘宝在 CDN 上面的实力，这在全世界都是数一数二的。另外因为 CDN 需要大量的服务器，要消耗很多能源（消耗多少？在前两年我们算过一笔帐，淘宝上产生一个交易，消耗的电足以煮熟 4 个鸡蛋）。这两年章文嵩的团队又在研究低功耗的服务器，在绿色计算领域也做了很多开创性的工作。淘宝 CDN 的发展需要专门一个章节来讲，想先睹为快的可以看一下笔者[对章文嵩的专访](http://qing.weibo.com/1866752224/6f4460e033000jme.html)。

　　回想起刚用缓存那段时间，笔者还是个小菜鸟，有一个经典的错误常常犯，就是数据库的内容更新的时候，忘记通知缓存系统，结果在测试的时候就发现我改过的数据怎么在页面上没变化呢。后来做了一些页面上的代码，修改 CSS 和 JS 的时候，用户本地缓存的信息没有更新，页面上也会乱掉，在论坛上被人说的时候，我告诉他用 Ctrl+F5 刷新页面，然后赶紧修改脚本文件的名称，重新发布页面。学会用 Ctrl+F5 的会员对我佩服的五体投地，我却惭愧的无地自容。

　　有些技术的发展是顺其自然的，有些却是突如其来的。到 2007 年的时候，我们已经有几百台应用服务器了，这上面的 Java 应用服务器是 WebLogic，而 WebLogic 是非常贵的，比这些服务器本身都贵。有一段时间多隆研究了一下 JBoss，说我们换掉 WebLogic 吧，于是又省下了不少银两。那一年，老马举办了第一届的“网侠大会”，会上来的大侠中有一位是上文提到的章文嵩，还有一位曾经在 JBoss 团队工作，我们也把这位大侠留下了，这样我们用起 JBoss 更加有底气了。

　　这些杂七杂八的修改，我们对数据分库、放弃 EJB、引入 Spring、加入缓存、加入 CDN、采用开源的 JBoss，看起来没有章法可循，其实都是围绕着提高容量、提高性能、节约成本来做的，由于这些不算大的版本变迁，我们姑且叫它 2.1 版吧，这个版本从构图上来看有 3 只脚，是不是稳定了很多？

　　架构图如下：

[](http://static.oschina.net/uploads/img/201411/13161406_c8yZ.jpg)

　　五、淘宝技术发展（Java时代：创造技术-TFS）

　　在讲淘宝文件系统 TFS 之前，先回顾一下上面几个版本。1.0 版的 PHP 系统运行了将近一年的时间（2003.05~2004.01）；后来数据库变成 Oracle 之后（2004.01~2004.05，叫 1.1 版本吧），不到半年就把开发语言转换为 Java 系统了（2004.02~2005.03，叫2.0版本）；进行分库、加入缓存、CDN之后我们叫它 2.1 版本（2004.10~2007.01）。这中间有些时间的重合，因为很多架构的演化并没有明显的时间点，它是逐步进化而来的。

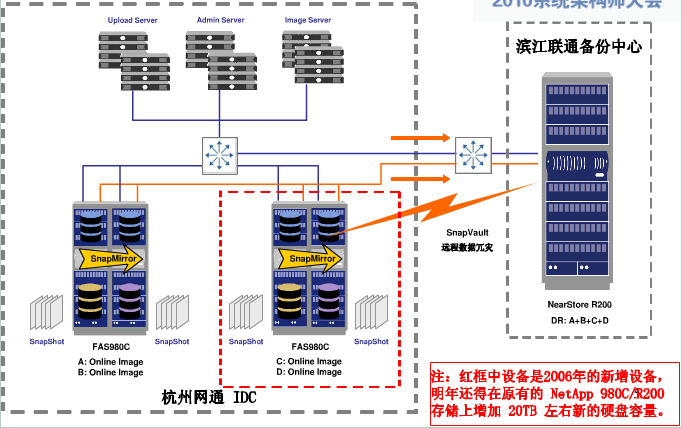
　　在描述 2.1 版本的时候我写的副标题是“坚若磐石”，这个“坚若磐石”是因为这个版本终于稳定下来了，在这个版本的系统上，淘宝网运行了两年多的时间。这期间有很多优秀的人才加入，也开发了很多优秀的产品，例如支付宝认证系统、招财进宝项目、淘宝旅行、淘宝彩票、淘宝论坛等等。甚至在团购网站风起云涌之前，淘宝网在 2006 年就推出了团购的功能，只是淘宝网最初的团购功能是买家发起的，达到卖家指定的数量之后，享受比一口价更低的价格，这个功能看起来是结合了淘宝一口价和荷兰拍的另一种交易模式，但不幸没有支撑下去。

　　在这些产品和功能的最底层，其实还是商品的管理和交易的管理这两大功能。这两大功能在 2.1 版本里面都有很大的变化。商品的管理起初是要求卖家选择 7 天到期还是 14 天到期，到期之后就要下架，必须重新发布才能上架，上架之后就变成了新的商品信息（ID变过了）。另外如果这个期间内成交了，之后再有新货，必须发布一个新的商品信息。这么做有几个原因，一是参照拍卖商品的时间设置，要在某日期前结束挂牌；二是搜索引擎不知道同样的商品哪个排前面，那就把挂牌时间长的排前面，这样就必须在某个时间把老的商品下架掉，不然它老排在前面；第三是成交信息和商品 ID 关联，这个商品如果多次编辑还是同一个 ID 的话，成交记录里面的商品信息会变来变去；还有一个不为人知的原因，我们的存储有限，不能让所有的商品老存放在主库里面。这种处理方式简单粗暴，但还算是公平。不过这样很多需求都无法满足，例如同样的商品，我上一次销售的时候很多好评都没法在下一个商品上体现出来；再例如我买过的商品结束后只看到交易的信息，不知道卖家还有没有再卖了。后来基于这些需求，我们在 2006 年下半年把商品和交易拆开。一个商家的一种商品有个唯一的 ID，上下架都是同一个商品。那么如果卖家改价格、库存什么的话，已成交的信息怎么处理？那就在买家每交易一次的时候，都记录下商品的快照信息，有多少次交易就有多少个快照。这样买卖双方比较爽了，给系统带来了什么？存储的成本大幅度上升了！

　　存储的成本高到什么程度呢？数据库方面提到过用了 IOE，一套下来就是千万级别的，那几套下来就是⋯⋯。另外淘宝网还有很多文件需要存储，我们有哪些文件呢？最主要的就是图片、商品描述、交易快照，一个商品要包含几张图片和一长串的描述信息，而每一张图片都要生成几张规格不同的缩略图。在 2010 年，淘宝网的后端系统上保存着 286 亿个图片文件。图片在交易系统中非常重要，俗话说“一张好图胜千言”、“无图无真相”，淘宝网的商品照片，尤其是热门商品，图片的访问流量是非常大的。淘宝网整体流量中，图片的访问流量要占到 90% 以上。且这些图片平均大小为 17.45 KB，小于 8K 的图片占整体图片数量 61%，占整体系统容量的 11%。这么多的图片数据、这么大的访问流量，给淘宝网的系统带来了巨大的挑战。众所周知，对于大多数系统来说，最头疼的就是大规模的小文件存储与读取，因为磁头需要频繁的寻道和换道，因此在读取上容易带来较长的延时。在大量高并发访问量的情况下，简直就是系统的噩梦。我们该怎么办？

　　同样的套路，在某个规模以下，采用现有的商业解决方案，达到某种规模之后，商业的解决方案无法满足，只有自己创造解决方案了。对于淘宝的图片存储来说，转折点在 2007 年。这之前，一直采用的商用存储系统，应用 NetApp 公司的文件存储系统。随着淘宝网的图片文件数量以每年 2 倍(即原来 3 倍)的速度增长，淘宝网后端 NetApp 公司的存储系统也从低端到高端不断迁移，直至 2006 年，即使是 NetApp 公司最高端的产品也不能满足淘宝网存储的要求。从 2006 年开始，淘宝网决定自己开发一套针对海量小文件存储的文件系统，用于解决自身图片存储的难题。这标志着淘宝网从使用技术到了创造技术的阶段。

　　2007年之前的图片存储架构如下图：

[](http://static.oschina.net/uploads/img/201411/13161407_J29d.png)

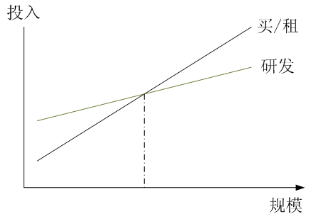
　　章文嵩博士总结了几点商用存储系统的局限和不足:

　　首先是商用的存储系统没有对小文件存储和读取的环境进行有针对性的优化；其次，文件数量大，网络存储设备无法支撑；另外，整个系统所连接的服务器也越来越多，网络连接数已经到达了网络存储设备的极限。此外，商用存储系统扩容成本高，10T的存储容量需要几百万，而且存在单点故障，容灾和安全性无法得到很好的保证。

　　谈到在商用系统和自主研发之间的经济效益对比，章文嵩博士列举了以下几点经验：

　　1. 商用软件很难满足大规模系统的应用需求，无论存储还是 CDN 还是负载均衡，因为在厂商实验室端，很难实现如此大的数据规模测试。

　　2. 研发过程中，将开源和自主开发相结合，会有更好的可控性，系统出问题了，完全可以从底层解决问题，系统扩展性也更高。

[](http://static.oschina.net/uploads/img/201411/13161407_cZk7.png)

　　3. 在一定规模效应基础上，研发的投入都是值得的。上图是一个自主研发和购买商用系统的投入产出比对比，实际上，在上图的交叉点左边，购买商用系统都是更加实际和经济性更好的选择，只有在规模超过交叉点的情况下，自主研发才能收到较好的经济效果。实际上，规模化达到如此程度的公司其实并不多，不过淘宝网已经远远超过了交叉点。

　　4. 自主研发的系统可在软件和硬件多个层次不断的优化。

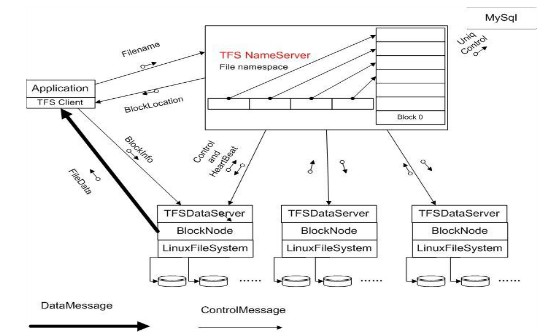
　　历史总是惊人的巧合，在我们准备研发文件存储系统的时候，Google 走在了前面，2007 年他们公布了 GFS（ Google File System ）的设计论文，这给我们带来了很多借鉴的思路。随后我们开发出了适合淘宝使用的图片存储系统TFS（Taobao File System）。3年之后，我们发现历史的巧合比我们想象中还要神奇，几乎跟我们同时，中国的另外一家互联网公司也开发了他们的文件存储系统，甚至取的名字都一样 —— TFS，太神奇了！（猜猜是哪家？）

　　2007 年 6 月，TFS 正式上线运营。在生产环境中应用的集群规模达到了 200 台 PC Server(146G\*6 SAS 15K Raid5)，文件数量达到上亿级别；系统部署存储容量：140TB；实际使用存储容量： 50TB；单台支持随机IOPS200+，流量 3MBps。

　　要讲 TFS 的系统架构，首先要描述清楚业务需求，淘宝对图片存储的需求大概可以描述如下：

　　文件比较小；并发量高；读操作远大于写操作；访问随机；没有文件修改的操作；要求存储成本低；能容灾能备份。应对这种需求，显然要用分布式存储系统；由于文件大小比较统一，可以采用专有文件系统；并发量高，读写随机性强，需要更少的 IO 操作；考虑到成本和备份，需要用廉价的存储设备；考虑到容灾，需要能平滑扩容。

　　参照 GFS 并做了适度的优化之后，TFS 1.0 版的架构图如下：

[](http://static.oschina.net/uploads/img/201411/13161407_3RqI.jpg)

　　从上面架构图上看：集群由一对 Name Server 和多台 Data Serve r构成，Name Server 的两台服务器互为双机，就是集群文件系统中管理节点的概念。

　　在这个架构中：  
　　• 每个 Data Server 运行在一台普通的 Linux 主机上  
　　• 以 block 文件的形式存放数据文件(一般64M一个block )  
　　• block 存多份保证数据安全  
　　• 利用 ext3 文件系统存放数据文件  
　　• 磁盘 raid5 做数据冗余  
　　• 文件名内置元数据信息，用户自己保存 TFS 文件名与实际文件的对照关系 – 使得元数据量特别小。

　　淘宝 TFS 文件系统在核心设计上最大的取巧的地方就在，传统的集群系统里面元数据只有 1 份，通常由管理节点来管理，因而很容易成为瓶颈。而对于淘宝网的用户来说，图片文件究竟用什么名字来保存实际上用户并不关心，因此TFS 在设计规划上考虑在图片的保存文件名上暗藏了一些元数据信息，例如图片的大小、时间、访问频次等等信息，包括所在的逻辑块号。而在元数据上，实际上保存的信息很少，因此元[**数据结构**](http://lib.csdn.net/base/datastructure)非常简单。仅仅只需要一个 fileID，能够准确定位文件在什么地方。

　　由于大量的文件信息都隐藏在文件名中，整个系统完全抛弃了传统的目录树结构，因为目录树开销最大。拿掉后，整个集群的高可扩展性极大提高。实际上，这一设计理念和目前业界的“对象存储”较为类似，淘宝网 TFS 文件系统已经更新到 1.3 版本，在生产系统的性能已经得到验证，且不断得到了完善和优化，淘宝网目前在对象存储领域的研究已经走在前列。

　　在 TFS 上线之前，淘宝网每个商品只允许上传一张图片，大小限定在 120K 之内，在商品详情里面的图片必须使用外站的服务。那时侯发布一件商品确实非常麻烦，笔者曾经想卖一台二手电脑，先把照片上传到 Google 相册，在发布到淘宝网之后发现 Google 相册被墙了，我的图片别人看不到，当时郁闷的不行。TFS 上线后，商品展示图片开放到 5 张，商品描述里面的图片也可以使用淘宝的图片服务，到现在为止，淘宝网给每个用户提供了 1G 的图片空间，这下大家都满足了。技术和业务就是这么互相用力的推动着，业务满足不了的时候，技术必须创新，技术创新之后，业务有了更大的发展空间。

　　1.3 版本的架构见阿里味（阿里巴巴内网）⋯⋯

　　六、淘宝技术发展（分布式时代：服务化）

　　在系统发展的过程中，架构师的眼光至关重要，作为程序员，把功能实现即可，但作为架构师，要考虑系统的扩展性、重用性，这种敏锐的感觉，有人说是一种代码洁癖。淘宝早期有几个架构师具备了这种感觉。一指开发的 Webx 是一个扩展性很强的框架，行癫在这个框架上插入了数据分库路由的模块、session 框架等等。在做淘宝后台系统的时候，同样需要这几个模块，行癫指导我把这些模块单独打成了 jar 包。另外在做淘宝机票、彩票系统的时候，页面端也有很多东西需要复用，最直观的是页头和页脚，一开始我们每个系统里面复制了一份过去，但奇妙的是，那段时间页脚要经常修改，例如把“雅虎中国”改成“中国雅虎”，过一段时间又加了一个“口碑网”，再过一段时间变成了“雅虎口碑”，最后又变成了“中国雅虎”，每个系统都改一遍，折腾啊。后来我就把这部分 velocity 模版单独拿出来了，做成了公用的模块。

　　上面这些都是比较小的复用模块，到 2006 年我们做了一个商品类目属性的改造，在类目里面引入属性的概念。项目的代号叫做“泰山”，如同它的名字，这是一个举足轻重的项目，这个改变是一个划时代的创新。在这之前的三年时间内，商品的分类都是按照树状的一级一级的节点来分的，随着商品数量的增长，类目也变得越来越深，越来越复杂，这带给买家的就是查找一件商品要逐级类目点开，找商品之前要懂商品的分类。而淘宝运营部门管理类目的小二也发现一个很严重的问题 —— 例如男装里面有T恤、T恤下面有耐克、耐克有纯棉的，女装里面也有T恤、T恤下面还是有耐克、耐克下面依然有纯棉的，那是先分男女装再分款式再分品牌再分材质呢？还是先分品牌再分款式再分材质再分男女呢？晕倒了。这时候，一位大侠出来了 —— 一灯，他说品牌、款式、材质这种东东可以叫做“属性”，属性是类似 tag 的一个概念，与类目相比更加离散，更加灵活，这样也缩减了类目的深度。这个思想的提出，一举解决了分类的难题！从系统的角度来看，我们建立了“属性”这样一个数据结构，由于除了类目的子节点有属性，父节点也有可能有属性，于是类目属性合起来也是一个结构化的数据对象。这个做出来之后我们把它独立出来作为一个服务，叫做 catserver（category server）。跟类目属性密切关联的商品搜索功能，独立出来，叫做 hesper（金星），catserver 和 hesper 供淘宝的前后台系统调用。

　　现在淘宝的商品类目属性已经是地球上最大的了，几乎没有什么类目的商品在淘宝上找不到（除了违禁的），但最初类目属性改造完之后，我们很缺属性数据，尤其是数码类的最缺。那从哪里弄这些数据呢亲？我们跟“中关村在线”合作，拿到了很多数据，那个时候，很多商品属性信息的后边标注着：“来自中关村在线”。有了类目属性，给运营的工作带来很大的便利，我们知道淘宝的运营主要就是类目的运营，什么季节推什么商品，都要在类目属性上面做调整，让买家更容易找到。例如夏天我要用户在女装一级类目下就标出来材质是不是蕾丝的、是不是纯棉的，冬天却要把羽绒衣调到女装一级类目下，流行什么就要把什么商品往更高级的类目调整。这样类目和属性要经常调整，随之而来的问题就显现了 —— 调整到哪个类目，那类商品的卖家就要编辑一次自己的商品，随着商品量的增长，卖家的工作量越来越大，然后我们就发现卖家受不了啦。到了 2008 年，我们研究了超市里面前后台商品的分类，发现超市前台商品可以随季节和关联来调整摆放场景（例如著名的啤酒和尿布的关联），后台仓库里面要按照自然类目来存储，二者密切关联却又相互分开。然后我们就把前后台类目分开了，这样卖家发布商品选择的是自然类目和属性，淘宝前台展示的是根据运营需要而摆放的商品的类目和属性。改造后的类目属性服务取名叫做 forest（森林，跟类目属性有点神似。catserver 还在，提供卖家授权、品牌服务、关键词等相关的服务）。类目属性的服务化，是淘宝在系统服务化方面做的第一个探索。

　　虽然个别架构师具备了代码洁癖，但淘宝前台系统的业务量和代码量还是爆炸式的增长了起来。业务方总在后面催，开发人员不够了就继续招人，招来的人根本看不懂原来的业务，只好摸索着在“合适的地方”加一些“合适的代码”，看看运行起来像那么回事，就发布上线了。在这样的恶性循环中，系统越来越臃肿，业务的耦合性越来越高，开发的效率越来越低。借用当时比较流行的一句话“写一段代码，编译一下能通过，半个小时就过去了；编译一下没通过，半天就过去了。”在这种情况下，系统出错的概率也逐步增长，常常是你改了商品相关的某些代码，发现交易出问题了，甚至你改了论坛上的某些代码，旺旺出问题了。这让开发人员苦不堪言，而业务方还认为这帮人干活越来越慢了。

　　大概是在 2007 年底的时候，研发部空降了一位从硅谷来的高管，空闻大师。空闻是一位温厚的长者，他告诉我们一切要以稳定为中心，所有影响系统稳定的因素都要解决掉。例如每做一个日常修改，都必须整个系统回归测试一遍；多个日常修改如果放在一个版本里面，要是一个功能没有测试通过，整个系统都不能发布。我们把这个叫做“火车模型”，任何一个乘客没有上车，都不许发车。这样做的最直接后果就是火车一直晚点，新功能上线更慢了，我们能明显的感觉到业务方的不满，空闻的压力肯定非常大。当时我都不理解这种一刀切的做法，为了稳定牺牲了发展的速度，这跟某 Party 的“稳定压倒一切”有什么分别？

　　但是到现在回过头来看看，其实我们并没有理解背后的思路。正是在这种要求下，我们不得不开始改变一些东西，例如把回归测试日常化，每天晚上都跑一遍整个系统的回归。还有就是在这种要求下，我们不得不对这个超级复杂的系统做肢解和重构，其中复用性最高的一个模块 —— 用户信息模块开始拆分出来了，我们叫它 UIC（user information center）。在 UIC 里面，它只处理最基础的用户信息操作，例如getUserById、getUserByName等等。

　　在另外一个方面，还有两个新兴的业务，也对系统基础功能的拆分提出了要求。在那个时候，我们做了淘宝旅行（trip.taobao.com）和淘宝彩票（caipiao.taobao.com）两个新业务，这两个新业务在商品的展示和交易的流程上都跟主站的业务不一样，机票是按照航班的信息展示的，彩票是按照双色球、数字和足球的赛程来展示的。但用到的会员的功能和交易的功能是跟主站差不多的，当时做的时候就很纠结，在主站里面做的话，会有一大半跟主站无关的东西，重新做一个的话，会有很多重复建设。最终我们决定不再给主站添乱了，就另起炉灶做了两个新的业务系统。从查询商品、购买商品、评价反馈、查看订单这一整个流程都重新写了一套出来。现在在“我的淘宝”里面查看交易记录的时候，还能发现“已买到的宝贝”里面把机票和彩票另外列出来了，他们没有加入到普通的订单里面去。在当时如果已经把会员、交易、商品、评价这些模块拆分出来，就不用什么都重做一遍了。

[](http://static.oschina.net/uploads/img/201411/13161407_D8tA.jpg)

　　到 2008 年初，整个主站系统（有了机票、彩票系统之后，把原来的系统叫做主站）的容量已经到了瓶颈，商品数在一亿以上，PV 在 2.5 亿以上，会员数超过了五千万。这个时候 Oracle 的连接池数量都不够用了，数据库的容量到了极限，上层系统再增加机器也无法继续扩容了，我们只有把底层的基础服务继续拆分，从底层开始扩容，上层才能扩展，这才能容纳以后三五年的增长。

　　于是那一年我们专门启动了一个更大的项目，把交易这个核心业务模块也拆分出来了。原来的淘宝交易除了跟商品管理耦合在一起，也在支付宝和淘宝之间跳来跳去，跟支付宝耦合在一起，系统复杂，用户体验也很不好。我们把交易的底层业务拆出来叫交易中心TC（trade center），所谓底层业务是例如创建订单、减库存、修改订单状态等原子型的操作；交易的上层业务叫交易管理TM（trade manager），例如拍下一件普通商品要对订单、库存、物流进行操作，拍下虚拟商品不需要对物流进行操作，这些在TM里面完成。这个项目取了一个很没有创意的名字 —— “千岛湖”，这帮开发人员取这个名字的目的是想在开发完毕之后，去千岛湖玩一圈，后来他们如愿以偿了。这个时候还有一个项目也在搞，就是淘宝商城，之前拆分出来的那些基础服务，给商城的快速构建，提供了良好的基础。

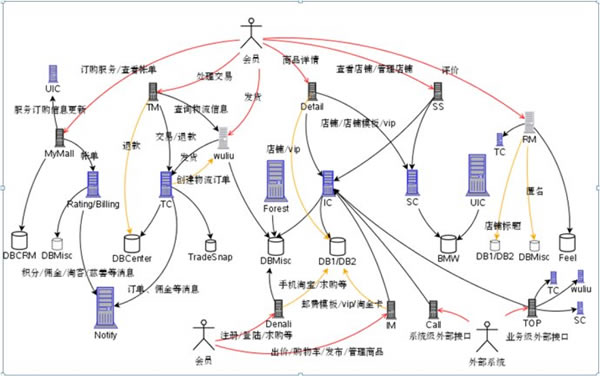
　　类目属性、用户中心、交易中心，随着这些模块逐步的拆分和服务化改造，我们在系统架构方面也积累了不少的经验。到 2008 年底干脆做了一个更大的项目，把淘宝所有的业务都模块化，这是继 2004 年从 LAMP 架构到 Java 架构之后的第二次脱胎换骨。这个项目取了一个很霸气的名字，叫“五彩石”（女娲炼石补天，用的石头）。这个系统重构的工作非常惊险，有人称之为“给一架高速飞行的飞机换发动机”。

　　五彩石项目发布之后，这帮工程师去三亚玩了几天。他们把淘宝的系统拆分成了如下架构：

[](http://static.oschina.net/uploads/img/201411/13161407_puQx.jpg)

　　其中 UIC 和 Forest 上文说过，TC、IC、SC分别是交易中心（Trade Center）、商品中心（Item Center）、店铺中心（Shop Center），这些中心级别的服务只提供原子级的业务逻辑，如根据ID查找商品、创建交易、减少库存等操作。再往上一层是业务系统TM（Trade Manager交易业务）、IM（Item Manager商品业务）、SM（Shop Manager，因为不好听，所以后来改名叫 SS：Shop System，店铺业务）、Detail（商品详情）。

　　拆分之后，系统之间的交互关系变得非常复杂，示意图如下：

[](http://static.oschina.net/uploads/img/201411/13161407_z7cs.jpg)

　　系统这么拆分的话，好处显而易见，拆分之后每个系统可以单独部署，业务简单，方便扩容；有大量可重用的模块以便于开发新的业务；能够做到专人专事，让技术人员更加专注于某一个领域。这样要解决的问题也很明显，分拆之后，系统之间还是必须要打交道的，越往底层的系统，调用它的客户方越多，这就要求底层的系统必须具有超大规模的容量和非常高的可用性。另外，拆分之后的系统如何通讯？这里需要两种中间件系统，一种是实时调用的中间件（淘宝的HSF，高性能服务框架）、一种是异步消息通知的中间件（淘宝的Notify）。另外还有一个需要解决的问题是用户在A系统登录了，到B系统的时候，用户的登录信息怎么保存？这又涉及到一个 Session 框架。再者，还有一个软件工程方面的问题，这么多层的一套系统，怎么去测试它？