目录

第一部分

淘宝技术发展1

引言：光棍节的狂欢

“时间到，开抢！”坐在电脑前早已等待多时的小美一看时间已到2011年11月11日零时，便迫不及待地投身于淘宝商城一年一度的大型网购促销活动——“淘宝双11购物狂欢节”。小美打开早已收藏好的宝贝——某品牌的雪地靴，飞快的点击购买、付款，一回头发现3000双靴子已被抢购一空。

小美跳起来，大叫一声“欧耶！”

小美不知道，就在11日零点过后的这一分钟里，全国有342万人和她一起涌入淘宝商城。当然，她更不知道，此时此刻，在淘宝杭州的一间办公室里，灯火通明，这里是“战时指挥部”，淘宝技术部的一群工程师正紧盯着网站的流量和交易数据。白板上是他们刚刚下的赌注，赌谁能最准确地猜中流量峰值和全天的交易总额。他们的手边放着充足的食物和各类提神的饮料。

一阵急促的电话铃声响起，是前线部门询问数据的，工程师大声报着：“第1分钟，进入淘宝商城的会员有342万人”。过了一会儿，工程师主动拿起电话：“交易额超过1亿元人民币了，现在是第8分钟。”接下来，“第21分钟，刚突破2亿元”，“第32分钟，3亿元了”，“第1个小时，4.39亿元”。这些数据随后出现在微博上，引起了一片惊呼。

“完蛋了！”突然有人大喝一声，所有的眼睛都紧张地盯着

他，只见他挠挠头，嘿嘿地笑道“我赌得少了，20亿元轻松就能过了，我再加5亿元。”他跑到白板边上把自己的赌注擦去，写上25，接下来有人写上28，有人写上30，有人到微博上开下盘口，同事们纷纷转载下注。接下来的这24个小时，战时指挥部的工程师们都不能休息，他们盯着网站的各种监控指标，适时调整机器，增减功能。顶住第一波高峰之后，这些人开始忙里偷闲地给自己买东西，大家互相交流着哪家买的移动硬盘靠谱，哪家衣服适合自己的女朋友，不时有人哀嚎宝贝被人抢了、信用卡额度不够了。同时，旁边白板上的赌注越下越大。

11月11日，这个棍子最多的日子被网民自我调侃地变成了一

个节日——“光棍节”，而淘宝网又用疯狂的折扣促销给它赋予了另外一个意义——“购物狂欢节”。2011年11月11日这一天，淘宝商城与淘宝网交易额之和突破52亿元人民币，这个数字是“购物天堂”香港一天零售总额8.5亿元的6倍。

网民感受到的是疯抢的喜悦，而网站的技术人员感受到的却是“压力山大”。就如同你家办酒席，宴请左邻右舍，这个办起来容易，倘若宴请十里八乡所有的人，吃饭的人固然开心，但却不是一般人家能够办得起来的。能办得起来如此盛宴者，需要强大的财力和物力、组织能力、技术实力（例如做这么多菜，你的炒锅一定要是“分布式的”、“可复制的”、“可扩展的”，洗菜和切菜要有“工作流引擎”，跑堂的要计算一下最优路径，甚至连厨房的下水道都要重新设计）。

淘宝能够举办如此盛宴，网站的技术实力可见一斑。至2011年年底，淘宝网拥有全国最大的Hadoop分布式计算集群之一（2000多个节点，CPU：24000 core，Memory：48000GB，Disk：24000块），日新增数据50TB，有40PB海量数据存储，分布在全国各地80多个节点的CDN网络，支撑的流量超过800Gbps。淘宝的搜索引擎能够对数十亿的商品数据进行实时搜索，另外，还拥有自主研发的文件存储系统和缓存系统，以及Java中间件和消息中间件系统，这一切组成了一个庞大的电子商务操作系统。从商业数据上看，Amazon的财报显示2011年完成了大约 480亿美元的交易额，eBay的2011年财报显示全年完成了大约600亿美元的交易额

（不包括其独立的汽车交易平台）。无论从交易额、商品数量还是从同比增速等指标上看，淘宝网均远超于此，是目前全球最大的电子商务平台。（由于淘宝是非上市公司，未公布2011年的业绩，以上内容来自淘宝网技术副总裁@_行癫 的微博）。

以上这些技术数据可能已经让一些人产生了不适的感觉，为了让更多的人读懂这本书，我们用下面这段文字描述一下小美访问淘宝网的时候，从技术的角度来看，网站上发生了什么样的事情。

参考资料来自《你刚才在淘宝上买了一件东西》（来自阿里员工卡特）

你发现快要过年了，于是想给你的女朋友买一件毛衣，你打开了www.taobao.com，这时你的浏览器首先查询DNS服务器，将www.taobao.com转换成IP地址。不过你首先会发现，在不同的地区或者不同的网络（电信、联通、移动）下，转换后的IP地址很可能是不一样的，这首先涉及到负载均衡的第一步，通过DNS解析域名时，将你的访问分配到不同的入口，同时尽可能保证你所访问的入口是所有入口中可能较快的一个（这和后文的CDN不一样）。

你通过这个入口成功地访问了www.taobao.com实际的入口IP地址，这时产生了一个PV（Page View，页面访问量。每日每个网站的总PV量是形容一个网站规模的重要指标。淘宝网全网在平日

（非促销期间）的PV大概是16～25亿个之间）。同时作为一个独立的用户，你这次访问淘宝网的所有页面均算作一个UV（Unique Visitor，用户访问）。最近臭名昭著的12306.cn的日PV量最高峰在10亿个左右，而UV量却远小于淘宝网十余倍，这其中的原因相信大家都知道。

因为同一时刻访问www.taobao.com的人数过于巨大，所以，即便是生成淘宝首页页面的服务器，也不可能仅有一台，仅用于生成www.taobao.com首页的服务器就可能有成百上千台，那么你的一次访问时生成页面给你看的任务便会被分配给其中一台服务器完成。这个过程要保证公正、公平、平均（即这成百上千台服务器每台负担的用户数要差不多），这一很复杂的过程由几个系统配合完成，其中最关键的便是LVS（Linux Virtual Server，世界上最流行的负载均衡系统之一，是由目前在淘宝网供职的章文嵩博士开发的）。

经过一系列复杂的逻辑运算和数据处理，这次用于给你看的淘宝网首页的HTML内容便成功生成了。对Web前端稍微有点常识的人都应该知道，浏览器下一步会加载页面中用到的CSS、JS（JavaScript）、图片等样式、脚本和资源文件。但是可能相对较少的人才会知道，你的浏览器在同一个域名下并发加载的资源数量是有限的，例如IE 6和IE 7是两个，IE 8是6个，chrome各版本不大一样，一般是4～6个。我刚刚看了一下，我访问淘宝网首页需要加载126个资源，那么如此小的并发连接数自然会加载很久。所

以前端开发人员往往会将上述这些资源文件分布在多个域名下，变相地绕过浏览器的这个限制，同时也为下文的CDN工作做准备。

据不可靠消息称，在2011年“双十一”当天高峰，淘宝的访问流量最巅峰达到871GB/s，这个数字意味着需要178万个4MB/s的家庭宽带才能负担得起，也完全有能力拖垮一个中小城市的全部互联网带宽。显然，这些访问流量不可能集中在一起，并且大家都知道，不同地区、不同网络（电信、联通等）之间互访会非常缓慢，但是你却很少发现淘宝网访问缓慢，这便是CDN（Content Delivery Network，即内容分发网络的作用）。淘宝在全国各地建立了数十个甚至上百个CDN节点，利用一些手段保证你访问的（这里主要指JS、CSS、图片等）站点是离你最近的CDN节点，这样便保证了大流量的分散已及在各地访问的加速。

这便出现了一个问题，那就是假若一个卖家发布了一个新的宝贝，上传了几张新的宝贝图片，那么淘宝网如何保证全国各地的CDN节点中都会同步存在这几张图片供用户使用呢？这就涉及大量的内容分发与同步的相关技术。另外，淘宝上拥有海量的宝贝图片等静态文件，这些文件的总容量也达到了数PB（1PB=1024TB=1048576GB），为了快速存取这些文件，淘宝开发了分布式文件系统TFS（TaoBao File System）来处理这类问题。

好了，这时你终于加载完成淘宝首页，然后习惯性地在首页搜索框中输入“毛衣”二字并按回车键，这时你又产生了一个

PV，然后，淘宝网的主搜索系统便开始为你服务，它首先对你输入的内容基于一个分词库进行分词操作。众所周知，英文是以词为单位的，词和词之间靠空格隔开，而中文是以字为单位，句子中所有的字连起来才能描述一个意思。例如，英文句子“I am a student”用中文表示，则为“我是一个学生”。计算机可以很简单地通过空格知道student是一个单词，但是不太容易明白“学”、“生”两个字合起来才表示一个词。把中文的汉字序列切分成有意义的词，就是中文分词，有些人也称为切词。“我是一个学生”分词的结果是“我 是 一个 学生”。

进行分词操作之后，还需要根据你输入的搜索词进行购物意图分析。用户进行搜索时常常有如下几类意图。

- 浏览型：没有明确的购物对象和意图，边看边买，用户比较随意和感性。Query[注1]例如：“2010年10大香水排行”、“2010年流行毛衣”、“zippo有多少种类？”；
- 查询型：有一定的购物意图，体现在对属性的要求上。Query例如：“适合老人用的手机”、“500元 手表”；
- 对比型：已经缩小了购物意图，具体到某几个产品。Query例如：“诺基亚E71 E63”、“akg k450 px200”；

注1　Query即查询。为了在数据库或搜索引擎中寻找某一特定文件、网站、记录或一系列记录，由搜索引擎或数据库送出的信息。——作者注

- 确定型：已经做了基本决定，重点考察某个对象。Query例如：“诺基亚N97”、“IBM T60”。

通过对你的购物意图的分析，主搜索会呈现出完全不同的结果。

之后的数个步骤后，主搜索系统便根据上述以及更多复杂的条件列出了搜索结果，这一切是由一千多台搜索服务器完成的。然后你开始逐一点击浏览搜索出的宝贝，查看宝贝详情页面。经常网购的亲们会发现，当你买过一个宝贝之后，即便是商家多次修改了宝贝详情页，你仍然能够通过“已买到的宝贝”查看当时的快照。这是为了防止商家对在商品详情中承诺过的东西赖账不认。显然，对于每年数十亿甚至上百亿笔交易的商品详情快照进行保存和快速调用不是一件简单的事情。这其中又涉及数套系统的共同协作，其中较为重要的是Tair（淘宝自行研发的分布式KV存储方案）。

接下来，无论你是否真的进行了交易，你的这些访问行为都会如实地被系统记录下来，用于后续的业务逻辑和数据分析。这些记录中的访问日志记录便是最重要的记录之一，但是从前面我们得知，这些访问是分布在各个地区多个不同的服务器上的，并且由于用户众多，这些日志记录都非常庞大，达到TB级别也非常正常。那么，为了快速、及时、同步地传输这些日志数据，淘宝研发了TimeTunnel，用于进行实时的数据传输，然后交给后端系

统进行计算报表等操作。

你的浏览数据、交易数据以及其他很多数据记录均会被保留下来，使得淘宝存储的历史数据轻而易举地便达到了数十甚至更多个PB。如此巨大的数据量存储在阿里巴巴集团的数据仓库中，并且其中有些数据使用了压缩比高达1:120的极限存储技术。之后这些数据会通过一个叫做云梯的基于Hadoop的由3000多台服务器组成的超大规模数据系统，以及一个基于阿里巴巴集团自主研发的ODPS系统的数据系统，不断地进行分析和挖掘。

淘宝从这些数据中能够知道小到你是谁，你喜欢什么，你的孩子几岁了，你是否在谈恋爱，喜欢玩魔兽世界的人喜欢什么样的饮料等，大到各行各业的零售情况、各类商品的兴衰消亡等海量的信息。

说了这么多，其实也只是叙述了淘宝上正在运行的成千上万个系统中的寥寥几个。即便是你仅仅访问一次淘宝的首页，所涉及的技术和系统规模都是你完全无法想象的，是淘宝2000多名顶级的工程师们的心血结晶，其中甚至包括长江学者、国家科学技术最高奖得主等众多牛人。同样，百度、腾讯等的业务系统也绝不比淘宝简单。你需要知道的是，你每天使用的互联网产品看似简单易用，背后却凝聚着难以想象的智慧与劳动。

（本文所涉及的技术与数据均来源于互联网）

为了有一个更直观的对比，我们说一个同行，他在2011年光棍节之前做促销，数据流量达到了12Gbps（他们有这么大的流量，老板很高兴，在微博上公布了这个数据），这时的流量达到了极限，网站几乎挂掉，用户无法下订单。而淘宝网光棍节当天网络的流量最高达到800Gbps，带给各家银行和快递公司的流量也让他们如临大敌（后来，他们以能够撑住淘宝带来的流量为荣而到处宣传）。另外，如果你在网上购买过火车票，更能体会到网站能支持多大的流量有多重要。但这不是一朝一夕就能做出来的，也不是有钱就能办到的。

以上对比的这些网站，也许读者很容易就能猜到是哪一家，这里绝对没有嘲笑他们的意思，采用通常的网站技术方案能做到这种程度已经不错了。任何网站的发展都不是一蹴而就的，通常是在什么阶段采用什么技术。在发展的过程中，网站会遇到各种各样的问题，正是这些原因才推动着技术的进步和发展，而技术的发展反过来又会促进业务的更大提升。二者互为因果，相互促进。如今淘宝网的流量已经是全球排名第12、国内排名第3（至2011年年底，eBay全球排名第20，国内前两名是百度和腾讯）。淘宝网的系统也从使用一台服务器，到采用万台以上的服务器。本书就为大家描述淘宝网在整个发展过程中，所有主动和被动的技术变革的前因后果，这由很多有趣的故事组成。

正如同很多人或组织成功了以后，就会为自己的出身编造一

个美丽的传说。关于淘宝网的出身，网上也有非常多的传说，下面我们就从它的出生开始讲起。

个人网站

LAMP

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集团，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。这里出一个问题考考大家，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?

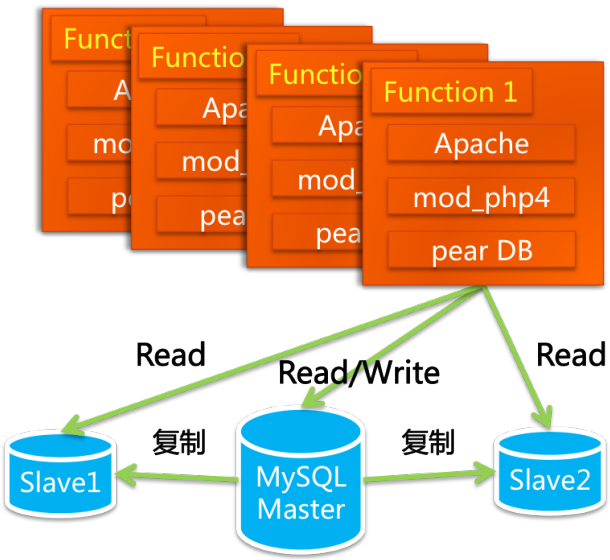
在说出这个答案之前，我们先介绍一下这个创业团队的成员：三个开发工程师（虚竹、三丰、多隆）、一个UED工程师（二当家）、三个运营工程师（小宝、阿珂、破天）、一个经理（财神），以及马云和他的秘书。当时对整个项目组来说，压

力最大的就是时间，为什么时间这么重要呢？火云邪神先生说过“天下武功无坚不破，唯快不破”，还有一个原因就是当时eBay和易趣在资本方面正打得不可开交，我们是趁虚而入的，等他们反应过来就危险了。那怎么在最短的时间内把一个网站从零开始建立起来呢？了解淘宝历史的人都知道淘宝是在2003年5月10日上线的，2003年4月7日到5月10日，这之间只有一个月时间。要是你在这个团队里，你怎么做？不是“抄一个来”，我们的答案是——“买一个来”。

买一个网站显然比做一个网站要省事，但是他们的梦想可不是做一个小网站而已，要做大，就不是随便买一个就行的，要有比较低的维护成本，要能够方便地扩展和二次开发。那么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的网站：LAMP（Linux+Apache+MySQL+PHP），这个直到现在还是一个很常用的网站架构模型，其优点是：无须编译，发布快速，PHP语言功能强大，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、免费的。

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（其官方网站 http://www.phpauction.netAuction，即是拍卖的意思，这个名字很直白，一眼就可看出这个系统是用什么语言做的、用途是什么），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，花了我们差不多2000美元（貌似现在降价了，只要946美元，在他们的网站上有明码标价的信息）。买来之后不是直接就能用的，需要很多本地化的修改，例如，修改一些数据类型，增加后台管理的功能，页面模板改得漂亮一点，页眉和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，现在把它拆分成一个主库、两个从库，并且读写分离。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得以提升（写要比读更加消耗资源，分开后互不干扰）。这样整个系统的架构就如下图所示。



其中，pear DB是一个PHP模块，负责数据访问层。另外，他们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，在当时，论坛几乎是所有网站的标配。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的功能（Admin系统），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，这是快速增加商品数的一个好方法）四种。（PHPAuction系统里只有拍卖的交易，Auction即拍卖的意思。@_行癫在微博中提到：今天，eBay所有的交易中，拍卖交易仍然占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基本上只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。）

在开发过程中，这个项目的代号是BMW（没错！就是宝马的意思）。这个是二当家提出的建议，二当家特别喜欢宝马，他希望我们的网站也如同宝马一样漂亮、快速、安全，充满乐趣。二当家现在的座驾就是一辆宝马X5，算是得偿所愿了。在上线的时候需要给这个网站取个名字，为了不引起eBay的注意，这个名字要撇开与阿里巴巴的关系，所以“阿里爷爷”、“阿里舅舅”之类的域名是不能用的。这时候，美女阿珂提供了一个很好听的名

字“淘宝”。因为她家里有人热爱收藏古董，经常去市场上淘宝贝，而她本人也非常热爱逛街，享受“淘”的乐趣，她觉得“淘宝”两个字特别符合网站的定位（阿珂说想到这个名字的时候，脑子里一道闪电劈过，真的是“灵光一闪”。后来“支付宝”的名字也是阿珂取的）。于是这个大名就定了下来，淘宝网横空出世了。

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，尤其是去类似商场等人多的地方。另外，在神州大地上最早出现的C2C网站易趣也正忙得不亦乐乎，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是收费的，为了收取交易佣金，eBay禁止买卖双方这么做，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，用户体验一下全变了，操作起来非常麻烦，很多易趣的卖家在那边都混不下去了，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，淘宝网的发展非常迅猛，2003年年底就吸引了注册用户23万个，每日31万个PV，从2003年5月到同年年底成交额达3371万元。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得真好！）

在市场和运营的后方，淘宝网的技术团队也在快速地做着系统的改进和创新。这里还有一个有趣的故事，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的昵称都选各种豆豆，例如“黄豆豆”、“蚕豆豆”、“黑眼豆豆”等。淘宝在讨论运营策略的时候提到这个问题，老马也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，我们用什么名字好呢？“淘淘”、“宝宝”、“淘宝宝”、“宝宝淘”？小宝都不满意，认为太女性化了，辱没了他堂堂一个七尺汉子。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，很有故事性和现场感。于是接下来很多武侠小说中的人物开始在淘宝论坛中行侠仗义，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、发布上线。公司给这几个人租了房子，他们合住在湖畔花园旁边的小区里（男女分开），每天睁开眼就去公司，半夜两三点收工睡觉，响应用户的需求非常快。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，据说，买个办公桌都要走两个月流程，反应速度自然慢了下来。

当时淘宝第一个版本的系统中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、我的淘宝等功能（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”，这是支付宝的雏形）。随着用户需求和流量的不断增长，系统做了很多日常改进，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WebApp。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），2003年7月，多隆又把阿里巴巴中文站的搜索引擎iSearch搬了过来。

如此快节奏的工作，其实大家都累得不轻，有人就提议大家

随时随地锻炼身体，可是当时SARS横行，在一个一百多平方米的房子里，怎么锻炼呢？高挑美女阿珂提议大家练习提臀操，这样有助于保持身材，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，与花名文化、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，这种存储引擎在写数据的时候会把表锁住。当Master同步数据到Slave的时候，会引起Slave写，这样在Slave的读操作都要等待。还有一点是会发生Slave上的主键冲突，经常会导致同步停止，这样，你发布的一些东西明明已经成功了，但就是查询不到。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）。

Oracle/支付宝/旺旺

讲到这里，顺便先辟个谣，网上有很多这样骗转发的励志段子：“1998年，马化腾等一伙人凑了50万元创办了腾讯，没买房；1998年，史玉柱借了50万元搞脑白金，没买房；1999年，丁磊用50万元创办了163.com，没买房；1999年，陈天桥炒股赚了50万元，创办盛大，没买房；1999年，马云等18人凑了50万元注册了阿里巴巴，没买房。如果当年他们用这50万元买了房，现在估计还在还着银行的贷款吧。”  其实上，阿里巴巴和淘宝网都是在

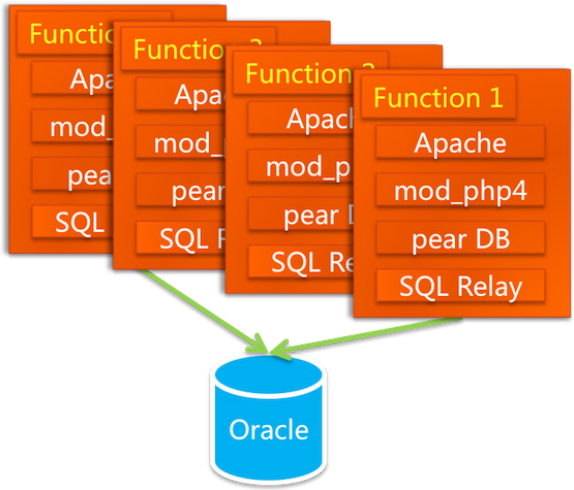
马云自己的房子里创办的，阿里巴巴是1999年初发布上线的。所以，关于马云买房子的事情，真相是这样的。

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推广运作。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。这时候，eBay终于看到淘宝网这个后起之秀了，他对竞争者的态度就是“封杀他”。eBay买断了新浪、搜狐、网易的电子商务类型的广告，签署了排他性协议，切断了淘宝在这上面做广告的路子。大路不通，我们就独辟蹊径，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，“淘宝网”几个字总是如影随形地出现在任何中小型网站上。市场部那位到处花钱买广告的家伙太能花钱了，一出手就是几百万元，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。换为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，有鲁国良、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（公布名单的网址为： http://apex.oracle.com/pls/otn/f?p=19297:3），当年全球只有十几名，而阿里巴巴就有4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，其访问方式和SQL语法都要跟着变，最重要的一点是，Oracle的性能和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，连接池中放的是长连接，是进程级别的，在创建进程的时候，它就要独占一部分内存空间。也就是说，这些连接数在固定内存的Oracle Server上是有限的，任何一个请求只需要从连接池中取得一个连接即可，用完后释放，这不需要频繁地创建和断开连接，而连接的创建和断开的开销是非常大的。但对于PHP语言来说，它对数据库的访问都是很直接的，每一个请求都要一个连接。如果是长连接，应用服务器增多时，连接数就多了，就会把数据库拖挂，如果是短连接，频繁地连接后再断开，性能会非常差（而Java语言有很多现成的连接池）。那如何是好呢？我们打探到eBay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，就放弃了找BEA的念头，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SQL Relay（http://sqlrelay.sourceforge.net），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，系统的架构就变成了如下形式：



数据一开始是放在本地的，七公带领的DBA们对Oracle做调优的工作，也对SQL进行调优。后来数据量变大后，本地存储无法满足了，买了NAS（Network Attached Storage，网络附属存储），NetApp（Network Appliance，美国网域存储技术有限公司）的NAS作为数据库的存储设备，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，性能一下提升了十几倍，这才比较稳定了。再后来，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，差点被辐射成蜘蛛侠。

替换完数据库后，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，他们在把数据的连接放在SQL Relay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，最快的解决办法就是“重启”它的服务。这在白天还好，只要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“SQL Relay进程挂起”的短信，就从春梦中醒来，打开电脑，连上机房的网络，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，干脆逃之夭夭。这是一个很严重的问题，一个人这么干了之后，很快就有更多的人学会了（这就是传说中的“病毒传播”）。然而魔高一尺，道高一丈，淘宝网这伙人开始研究防骗子的解决方案，他们看了PayPal的支付方式，发现不能解决问题。研究了类似QQ币的东西，想弄个“淘宝币”出来，发现也不行。后来这几个聪明的脑袋把这些想法糅合起来，突然想

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，淘宝网上线了一个功能，叫做“安全交易”，卖家如果选择支持这种功能，买家就会把钱交给淘宝网，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”。这个功能最早是让卖家可选的，因为这会延迟他收款的周期。但一旦卖家用了这个之后，就发现交易量猛增，一年之后，几乎所有的卖家都选择担保交易，到后来干脆所有的交易都必须走担保交易。在2012年支付宝的年会上，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。

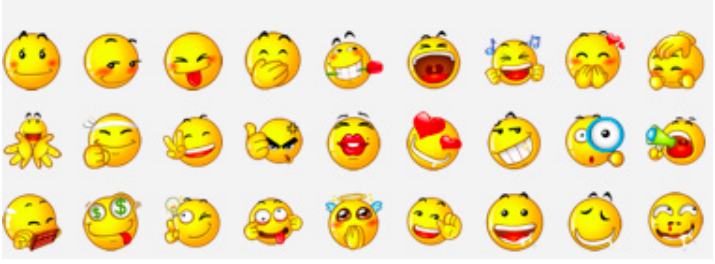
当时开发“安全交易”功能的是茅十八和他的徒弟苗人凤（茅十八开发到一半去上海读MBA了，苗人凤现在是支付宝的首席业务架构师），开发与银行网关对接功能的是多隆。当时多数银行的网站已经支持在线支付了，但多隆告诉我，他们的网关五花八门，用什么技术的都有，我们必须一家一家地去接。而且银行的网关不保证用户付钱后就一定能扣款成功，不保证扣款成功后就一定通知淘宝，也不保证通知淘宝后就一定能通知到，以及不保证通知到了就不重复通知。这害苦了苗人凤，他必须每天手工核对账单，少一分钱都睡不着觉，因为对不齐账就一定是有人的钱找不到地方了，这可是天大的问题。另外，他为了测试这些功能，去杭州所有的银行都办理了一张银行卡。一大堆银行卡摆在桌子上，不知道的人还以为他一定很有钱（高富帅啊），其

实里面都只是十元八元的。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。如果说和易趣过招的第一招是免费，这让用户无须成本就能进来，那么第二招就是“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，他们不会局限于一两招，一旦出手，则连绵不绝。而淘宝的第三招就是“旺旺”。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，便宜5毛行不？”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，我就深刻地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”。最早做旺旺开发的人只有一个——无崖

子，我们叫他“旺旺之父”，为了支持他的工作，我们工作用的IM工具仅限于旺旺，旺旺在线数上新高之后，他请我们吃鸭脖子。有时候为了吃到鸭脖子，我们盯着在线数看，快到整数量的时候，自己赶紧去挂几个小号上去。还有一个很卡哇伊的设计师MM——奇灵，开发出了一套旺旺表情，这套表情比所有的聊天软件的表情都大，也更加生动，一直沿用到现在，我们叫奇灵为“旺旺之母”。



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。

SQL Relay的问题搞得三丰等人很难睡个囫囵觉，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。但不能总这样通过不断地重启来保证系统的稳定性。于是，2004年上半年开始，整个网站就开始了一个脱胎换骨的手术。

第二部分

淘宝技术发展2 

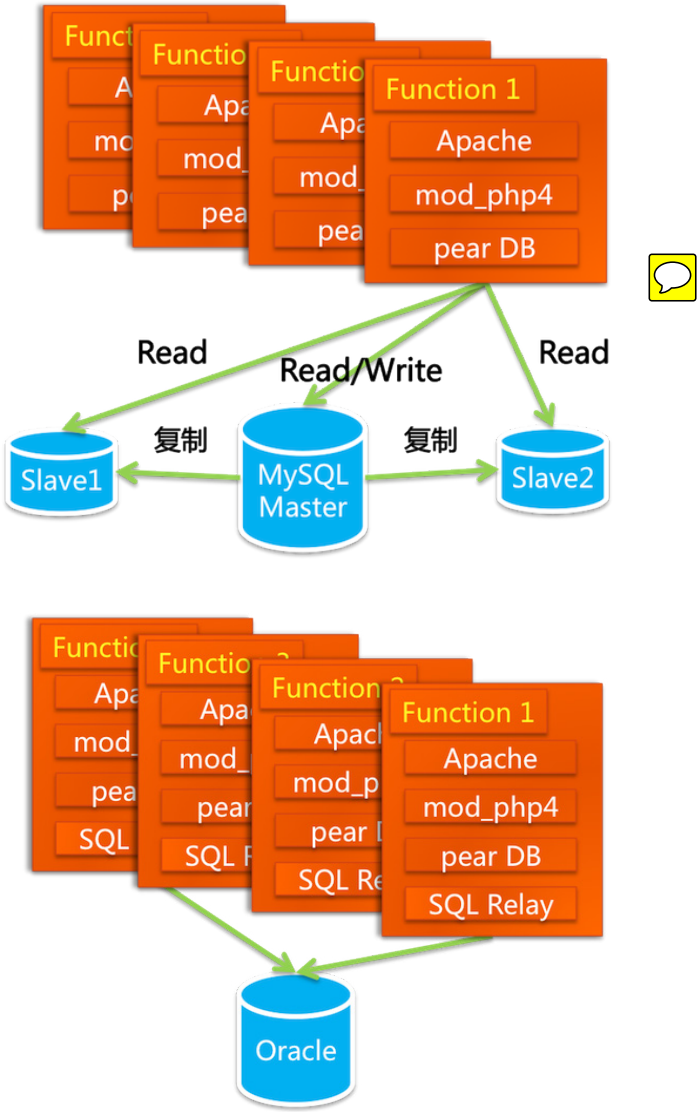
Java时代

脱胎换骨

我的师父黄裳曾经说过“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出来。后来我看了很多系统的架构，发现这个言论基本成立。反观淘宝以前两个版本的架构，如下图所示，你看哪个比较美？

显然，第一个比较好看，第二个显得头重脚轻，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，那么从哪里动刀呢？只有换开发语言了。换什么语言好？用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地增加。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。现在再考一下大家：亲，如果你在这个创业团队中，请什么样的人来做这件

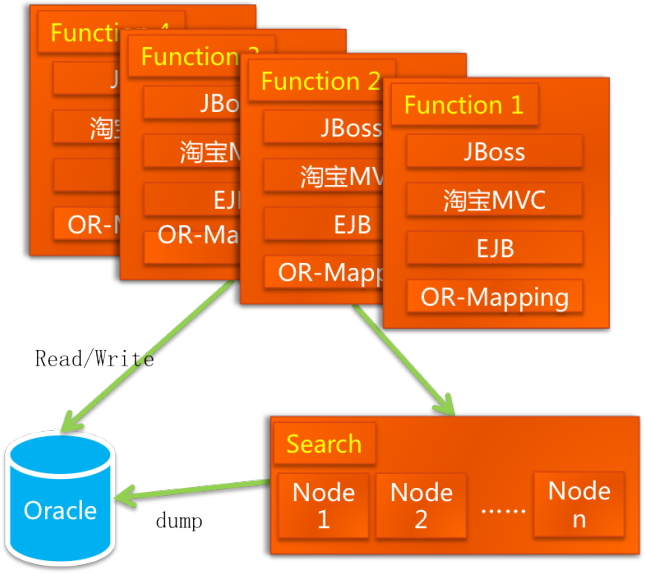

Function 1
Apache
mod_php4
pear DB
Read
Read/Write
Read
复制
复制
Slave1
MySQL
Master
Slave2
Function 1
Apache
mod_php4
pear DB
SQL Relay
Oracle

事？我们的答案是请Sun公司的人。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，我刚才说到Java被世界上主流的大规模网站普遍采用，其中有一个网站就是eBay，那时侯eBay的系统刚刚从C++ 改到Java，而且就是请Sun的工程师给改造成Java架构的，这下你懂了吧？他们不仅更懂Java，而且更懂eBay。

Sun公司的这帮工程师的确很强大，在笔者2004年年底来淘宝的时候，他们还在，我有幸与他们共事了几个月。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，“too young, 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，一个模块一个模块地渐进式替换。如用户模块，老的member.taobao.com继续维护，不添加新功能，新功能在新的模块上开发，跟老的模块共用一个数据库，开发完毕之后放到不同的应用集群上，另开一个域名member1.taobao.com，同时再替换老的功能，替换一个，就把老的模块上的功能关闭一个，逐渐把用户引导到member1.taobao.com，等所有的功能都替换完之后，关闭member.taobao.com上。从设计上来看，这个member1的

二级域名应该是一个过渡状态，但我们把member域名的代码下线以后，发现很难把member1切换回member，因为有些地方把链接写死了，于是后来很长时间里我们都是在用member1.taobao.com这样奇怪的域名。一年后，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，复制得毫无保留，我们只能会心一笑。

说了开发模式，再说说用到的Java MVC框架，当时的struts1.x是用得比较多的框架，但是用过webwork和struts2的人可能知道，struts1.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，有个架构师周悦虹，他在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等，控制层可以用EJB和Spring（Spring是后来才有的）。项目组选择了这个强大的框架。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西用不到，但他们还是极力坚持。在经历了很多次的技术讨论、争论甚至争吵之后，这个系统的架构就变成了下图的形式。



MVC框架是阿里的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中。这个架构图是不是好看了一点了？

Sun的这帮工程师开发完淘宝的网站之后，用同样的架构又做了一个很牛的网站，叫“支付宝”。（上一篇说过支付宝最初是淘宝上的“安全交易”功能，这个功能后来独立出来，成立了一个网站，也成立了一个公司，就是现在的支付宝。把支付宝从淘宝分出去的人，就是Sun公司的这几个人。）下图是支付宝的第一次员工大会。

上面的架构中，引入了搜索引擎iSearch（前文说过，iSearch其实是在LAMP系统运行一段时间之后被多隆引进的，换为Oracle之后只是替换一下数据源）。其实这个搜索引擎的原理很简单，就是把数据库里的数据dump（倾倒）成结构化的文本文件后，放在硬盘上，提供Web应用以约定的参数和语法来查询这些数据。这看起来不难，难的是数以亿计的信息，怎么做到快速更新呢？这好比你做了一个网站，在百度上很快就能搜到，你一定很满意了。但如果你发布一件商品，在淘宝上过1个小时还搜不到，你肯定要郁闷了。另一个难点是如何保证非常高的容量和并发量？再往后面就要考虑断句和语义分析的问题，以及推荐算法等更加智能的问题。这些内容先不详细介绍，因为搜索引擎的技术已经足以写好几本书了。

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存在于数据和存储方面。前面也说到，MySQL撑不住之后换为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，再后来，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。在2004年夏天，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欢迎他们，因为那是我们最值钱的设备，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就运行在了小型机上，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，一级一级地往上跳。

到2004年底，淘宝网已经有4百多万种商品了，日均4千多万个PV，注册会员达400万个，全网成交额达10亿元。

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。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1亿个PV的网站，但淘宝网的发展实在是太快了，到了10亿个PV怎么办？到了百亿怎么办？在几年以后，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

步地把IOE（IBM小型机、Oracle、EMC存储）这几个“神器”都去掉了。这些神器就如同《西游记》中那些神仙的兵器，他们身边的妖怪们拿到这些兵器能把猴子打得落荒而逃。但最牛的神仙是不依赖这些神器的，他们挥一挥衣袖、翻一下手掌就威力无比了。

坚若磐石

已经有读者在迫不及待地问怎么去掉了IOE？别急，在去掉IOE之前还有很长的路要走（在后面讲到TDDL的时候，会提到去IOE的一些事情）。行癫等人买回小型机之后，我们用上了

Oracle。然后七公带着一帮DBA做优化SQL和存储方面的工作，行癫带着几个架构师研究数据库的扩展性。Oracle本身是一个封闭的系统，用Oracle怎么做扩展呢？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，查询速度与容量成反比。简单地说，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直接加机器就可以了。我们的做法是把用户的信息按照ID来存放到两个数据库中（DB1和DB2），把商品的信息和卖家信息放在两个对应的数据库中，把商品类目等通用信息放在第三个库中（DBcommon）。这么做的目的是除了增加了数据库的容量之外，还有一个就是做容灾，即万一一个数据库挂了，整个网站上还有一半的商品可以买。

数据库这么分后，应用程序就会出现麻烦，如果我是卖家，查看我的商品没有问题，我们都在一个库里。但如果我是一个买家，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？必须到两个数据库中分别查询对应的商品。要按时间排序怎么办？两个库中“我已买到的宝贝”全部查出来在应用程序中做合并。另外，分页怎么处理？关键字查询怎么处理？专业点的说法就是数据的Join没法做了。这些工作交给程序员来做也许会更麻烦，于是行癫出手了，他写了一个

数据库路由的框架DBRoute，统一处理了数据的合并、排序、分页等操作，让程序员像使用一个数据库一样操作多个数据库里的数据，这个框架在淘宝的Oracle时代一直在使用。但是后来随着业务的发展，这种分库的第二个目的——“容灾”的效果没有达到。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，在他们离开之后，EJB也逐渐被冷落了下来。在2005年和2006年的时候，Spring大放异彩，于是在控制层用Spring替换掉了EJB，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1663万个，PV有8931万个，注册会员有1390万个，这给数据存储带来的压力依然很大，数据量大，速度就慢。亲，除了搜索引擎、分库分表，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，9000万次的访问量，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（全部从数据库中读出来，不写入数据库），在那个时候，我们的架构师多隆大神做了一个基于 Berkeley DB 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，我们并不敢把所有能缓存的信息都往里面放，一开始先把卖家的信息放里面，然后把商品属性放里面，再把店铺信息放里面，但是像商品详情这类字段太大的放进去受不了。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，在系统里其实放在哪里都不招人待见。笔者清楚地记得，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中移出来。它最早与商品的价格、运费等信息放在一个表中，拖慢了整张表的查询速度，而很多时候查询商品信息是不需要查看详情的。于是在2005年的时候，我把商品详情放在数据库的另外一张表中，再往后，这个大字段被从数据库中请了出来，先是放入了缓存系统，到现在是放进了文件系统TFS中。

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”（这个信息是2006年加上去的），这个数字每刷新一次，页面就要“写入”存储一次，这种高频度实时更新的数据能用缓存吗？通常来说，这种是必须放进数据库的，但是悲剧的是，我们在2006年开发这个功能的时候，把浏览量写入数据库，发布上线1个小时后，数据库就挂掉了，每天几亿次的写入，数据库承受不了。那怎么办？亲，……先不回答你，后面讲到缓存Tair的时候再说。（下图不是广告，请把注意力从左边移到右边中间，看看浏览量这个数据在哪里。）

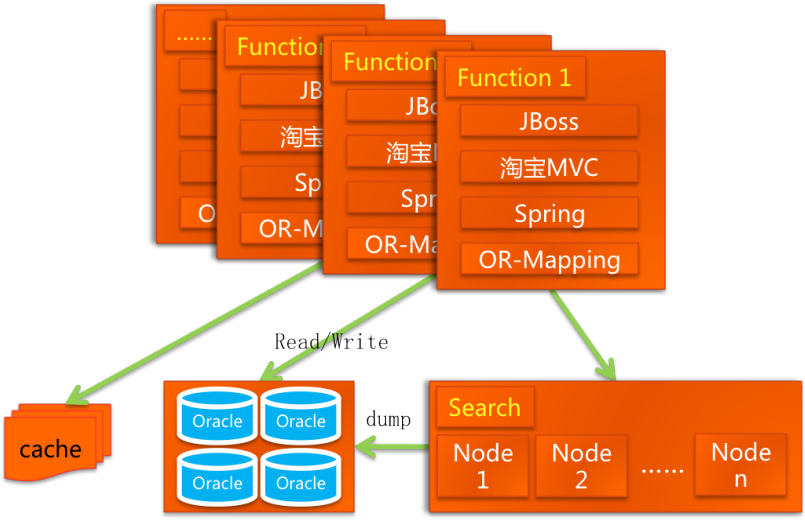
CDN这个工作相对比较独立，跟别的系统一样，一开始我们采用的也是商用系统。后来随着流量的增加，商用的系统已经撑不住了，LVS的创始人章文嵩博士带人搭建了淘宝自己的CDN网络。在本文的引言中，我说过淘宝的CDN系统支撑了800Gbps以

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。”——淘宝一家的流量比他们的加起来还要多，这样你可以看出淘宝在CDN上的实力，这在全世界都是数一数二的（其实我们一开始用的商用CDN就是ChinaCache，它们支撑了很长时间）。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账，淘宝上产生一个交易，消耗的电量足以煮熟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，我们定制的基于英特尔凌动处理器的低功耗服务器已经部署到了CDN机房，降低了很大的能耗。这方面的内容可以看后面笔者对章文嵩博士的专访。

回想起刚用缓存那段时间，笔者还是个菜鸟，有一个经典的错误常常犯，就是更新数据库的内容时，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在页面上没有变化。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组合键（清除本地缓存刷新页面），然后赶紧修改脚本

文件的名称，重新发布页面。

我们对数据分库、放弃EJB、引入Spring、加入缓存、加入CDN等工作，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，由于这些是不算大的版本变迁，我们姑且叫它2.1版，这个版本从构图上看有三只脚，是不是稳定了很多？



在这个稳定的版本下，淘宝网的业务有了突飞猛进的发展，2005年5月，微软的MSN门户大张旗鼓地进入中国，淘宝网成为它的购物频道。2005年中，盛大进军机顶盒业务，其电视购物的功能也是淘宝网开发的。虽然因为水土不服或者政策的原因，这

两个业务现在都看不到了，但他们曾经是中国互联网行业的大事件。那位和微软谈合作的人是@胖胡斐，他花起钱来也是大手笔的，我们就管他叫“二少爷”，他现在是蘑菇街的COO。

另外，老马也从来都不缺少娱乐精神，他看到湖南卫视的超女如此成功，也想借鉴一下这种模式。就在2005年底，淘宝网和湖南卫视合作推出了一档节目，叫做“超级Buyer秀”。这是一个定位于“超级会网购的人”的一个选秀节目，以百万年薪为诱饵，让大家分享自己的网购经历，网友投票选出最终胜者。这个从海选到表演，历时一年多，广告做得铺天盖地。虽然节目最终没有超女那样火爆，这也让“淘宝网就是网购”的形象通过湖南卫视更加深入人心。

到2006年，淘宝网已经有了1.5亿个的日均PV，商品数达5千多万个，注册用户3千多万个，全网成交额达169亿元。

创造技术

TFS

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年5月—2004年1月），服务器由一台发展到多台；

后来数据库撑不住了，将MySQL换成了Oracle，引入了搜索引擎（2004年1月—2004年5月，叫1.1版本）；然后不到半年的时间又把开发语言换成了Java（2004年2月—2005年3月，叫2.0版本），数据服务逐步采用了IOE；随着数据量和访问量的增长，我们进行数据分库、加入缓存、使用CDN（2004年10月—2007年1月，叫2.1版本）。这几个版本中间有些时间上的重合，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，商品的类目属性、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，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。

在这些产品和功能的最底层，其实还是商品管理和交易管理这两大功能。这两大功能在2.1版本中都有很大的变化。商品管理起初是要求卖家选择7天到期还是14天到期，到期之后自动下架，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，如果商品在这个期间成交了，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

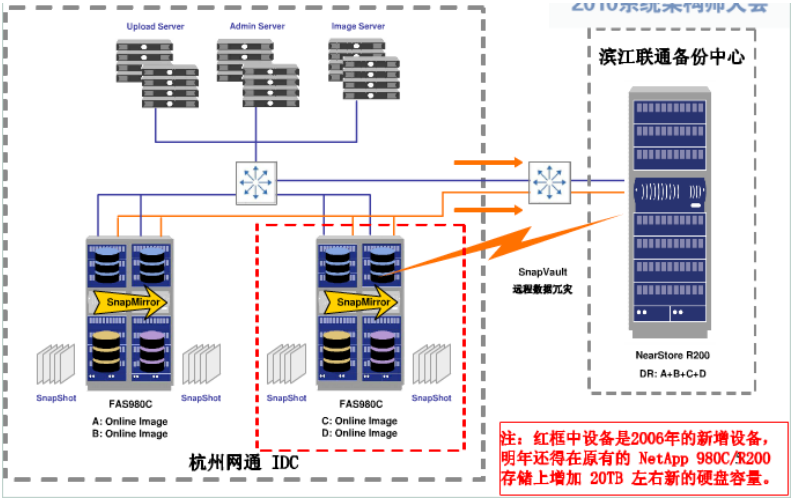
样的商品哪个排在前面，那就把挂牌时间长的排前面（这样就必须在某个时间把老的商品下架，否则它会一直排在前面）；第三是成交信息和商品ID关联，这个商品如果多次编辑还是同一个ID的话，成交记录中的商品信息会不断改变；还有一个不为人知的原因是我们的存储有限，不能让所有的商品老存放在主库中。这种处理方式简单粗暴，但还算是公平。不过这样会导致很多需求都无法满足，例如，卖出一件商品之后就无法更改价格，否则前面已经成交的那个价格都变了，而且同样的商品，上一次销售后的很多好评都无法在下一个商品上体现出来；再如，我买过的商品结束后只看到交易的信息，不知道卖家是否还会卖。基于这些需求，我们在2006年下半年把商品和交易拆开，一个商家的一种商品有一个唯一的ID，上下架都是同一个商品。那么如果卖家修改价格和库存等信息，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，但这给系统带来了什么？存储的成本大幅度上升了！

存储的成本高到什么程度呢？数据库方面用了IOE，一套下来就是千万级别的，那几套下来就是——好多钱啊。另外，淘宝网还有很多文件需要存储，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，大

家常说“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。在淘宝网整体流量中，图片的访问流量要占到90%以上，而且这些图片平均大小为17.45KB，小于8KB的图片占整体图片数量的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。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，我们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所示。



在一次架构师大会上，章文嵩博士总结了几点商用存储系统的局限和不足。

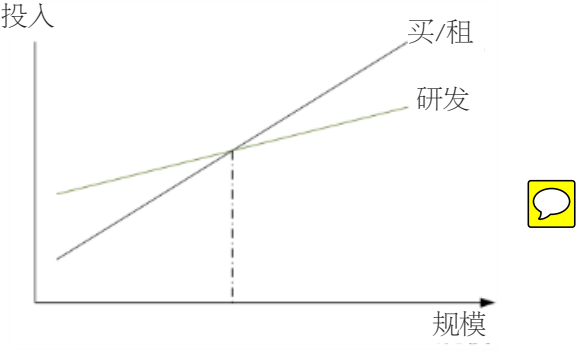
第一，商用存储系统没有对小文件存储和读取的环境进行有针对性的优化；第二，文件数量大，网络存储设备无法支撑；第三，整个系统所连接的服务器越来越多，网络连接数已经达到网络存储设备的极限；第四，商用存储系统扩容成本高，10TB的存储容量需要几百万元，而且存在单点故障，容灾和安全性无法得到很好的保证。

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

第一，商用软件很难满足大规模系统的应用需求，无论是存储、CDN还是负载均衡，在厂商实验室端，都很难实现如此大的数据规模测试。

第二，在研发过程中，将开源和自主开发相结合，会有更好的可控性，若系统出了问题，完全可以从底层解决问题，系统扩展性也更高。

第三， 在一定规模效应的基础上，研发的投入都是值得的。下图演示的是一个自主研发和购买商用系统的投入产出比，实际上，图中交叉点的左边，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并不多，不过淘宝网已经远远超过了交叉点。



第四，自主研发的系统可在软件和硬件的多个层次之间不断

优化。

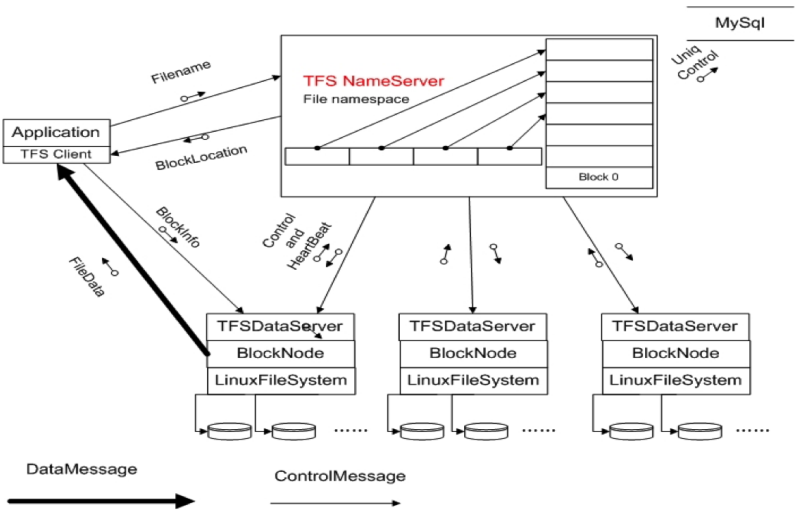
历史总是惊人的巧合，在我们准备研发文件存储系统的时候，Google走在了前面，2007年，他们公布了GFS（Google File System）的设计论文，这给我们带来了很多借鉴的思路。随后我们开发出了适合淘宝使用的图片存储系统（TaoBao File System，TFS）。3年之后，我们发现历史的巧合比我们想象的还要神奇，几乎跟我们同时，中国的另外一家互联网公司也开发了他们的文件存储系统，甚至取的名字都一样——TFS，太神奇了！（猜猜是哪家）

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（146GB×6 SAS 15KB Raid5），文件数量达到上亿级别；系统部署存储容量为140TB；实际使用存储容量为50TB；单台支持随机IOPS 200+，流量为3MB/s。

说到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。显然，应对这种需求时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；由于并发量高，读写随机性强，需要更少的I/O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并做了大量的优化之后，TFS 1.0版的架构图如下。



从上面的架构图可看出：集群由一对Name Server和多台Data Server构成，Name Server的两台服务器互为双机，这就是集群文件系统中管理节点的概念。

在这个系统中：

- 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（一个Block的大小一般为64MB）；
- Block存储多份是为了保证数据安全；

- 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，这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在于，传统的集群系统中元数据只有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他们并不关心，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、访问频次等信息，包括所在的逻辑块号。而在实际的元数据上，保存的信息很少，因此，元数据结构非常简单。仅仅只需要一个FileID就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可极大地提高。实际上，这一设计理念和目前业界的“对象存储”较类似。

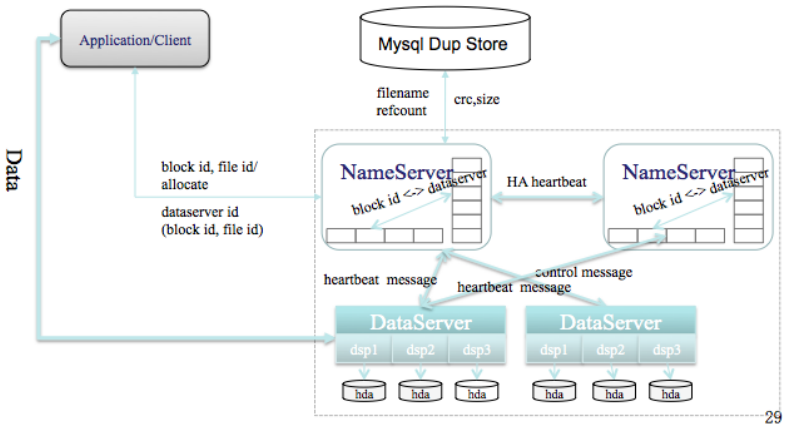
在TFS上线之前，淘宝网每个商品只允许上传一张图片，大小限定在120KB之内，在商品详情中的图片必须使用外站的服务。那时侯发布一件商品确实非常麻烦，笔者曾经想卖一台二手电脑，我先把照片上传到Google相册，在发布到淘宝网之后发现Google相册被墙（即被屏撇，无法访问），我的图片别人看不

到，当时很郁闷。在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1GB的图片空间。技术和业务就是这么互相借力推动着的，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。

TFS发布之后，又经历了多个版本的修改，到1.3版的时候已经比较成熟了。2009年6月，TFS 1.3版本上线，集群规模大大扩展，部署到淘宝的图片生产系统上，整个系统已经从原有200台PC服务器扩增至440台PC服务器（300B×12 SAS 15KB RPM）+30台PC服务器（600B×12 SAS 15KB RPM）。支持文件数量也扩容至百亿级别；系统部署存储容量为1800TB；当前实际存储容量为995TB；单台DataServer支持随机IOPS900+，流量为15MB+；

目前NameServer运行的物理内存是217MB（服务器使用千兆网卡）。

TFS 1.3版本逻辑结构图如下图所示。



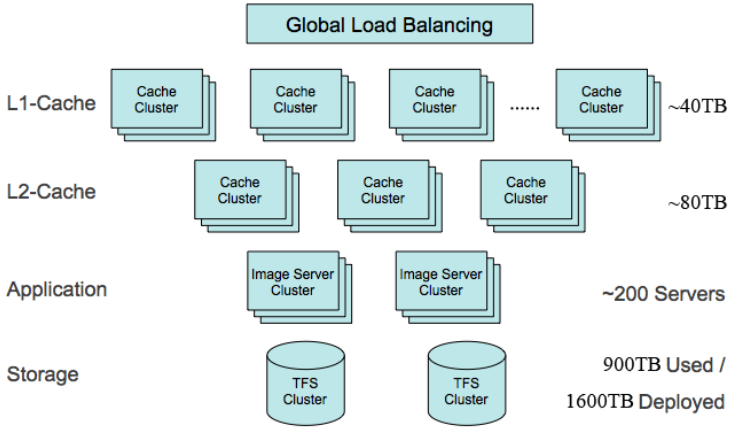
在TFS 1.3版本中，工程师们重点改善了心跳和同步的性能，最新版本的心跳和同步在几秒钟之内就可完成切换，同时进行了一些新的优化，包括元数据存储在内存中、清理磁盘空间等。

性能上也做了优化，包括如下内容。

- 采用ext4文件系统，并且预分配文件，减少ext3等文件系统数据碎片带来的性能损耗；
- 单进程管理单块磁盘的方式，摒除RAID5机制；

- 带有HA机制的中央控制节点，在安全稳定和性能复杂度之间取得平衡；
- 缩减元数据大小，将更多的元数据加载入内存，提升访问速度；
- 跨机架和IDC的负载均衡及冗余安全策略；
- 完全平滑扩容。

对于整个图片服务来说，仅有TFS还是不够的，整个图片服务机器的拓扑结构如下图所示。



整个图片存储系统就像一个庞大的服务器，有处理单元、缓存单元和存储单元。前面已经介绍过后台的TFS集群文件存储系

统，在TFS前端，还部署着200多台图片文件服务器，用Apache实现，用于生成缩略图的运算。

值得一提的是，根据淘宝网的缩略图生成规则，缩略图都是实时生成的。这样做的好处有两点：一是为了避免后端图片服务器上存储的图片数量过多，大大节约后台存储空间的需求，我们计算过，采用实时生成缩略图的模式比提前全部生成好缩略图的模式节约90%的存储空间。也就是说，存储空间只需要后一种模式的10%。二是，缩略图可根据需要实时生成，这样更加灵活。

图片文件服务器的前端则是一级缓存和二级缓存，前面还有全局负载均衡的设置，用于解决图片的访问热点问题。图片的访问热点一定存在，重要的是，让图片尽量在缓存中命中。目前淘宝网在各个运营商的中心点设有二级缓存，整体系统中心店设有一级缓存，加上全局负载均衡，传递到后端TFS的流量就已经非常均衡和分散了，对前端的响应性能也大大提高。根据淘宝的缓存策略，大部分图片都尽量在缓存中命中，如果缓存中无法命中，则会在本地服务器上查找是否存有原图，并根据原图生成缩略图，如果都没有命中，则会考虑去后台TFS集群文件存储系统上调取。因此，最终反馈到TFS集群文件存储系统上的流量已经被大大优化了。

淘宝网将图片处理与缓存编写成基于Nginx的模块，我们认为，Nginx是目前性能最高的HTTP服务器（用户空间），代码清

晰，模块化非常好。淘宝网使用GraphicsMagick进行图片处理，采用了面向小对象的缓存文件系统，前端有LVS+Haproxy将原图和其所有的缩略图请求都调度到同一台Image Server（图片服务器）。

在文件定位上，内存用Hash算法做索引，最多一次读盘。另外会有很多相同的图片重复上传上来，去除重复文件也是采用Hash算法来做的。写盘方式则采用Append方式写，并采用了淘汰策略FIFO，主要考虑降低硬盘的写操作，没有必要进一步提高Cache命中率，因为ImageServer和TFS位于同一个数据中心，读盘效率还是非常高的。

目前淘宝网的TFS已经开源（见code.taobao.org），业界的同仁可以一起使用和完善这个系统。

Tair

TFS的开发让淘宝的图片功能得到了充分的发挥。同TFS一样，很多技术都是在产品的推动下得到发展的。在介绍下面的技术之前，有必要说说前些年我们做过的几个产品。

先说一个比较悲剧的——“团购”，这个团购可不是现在满大街挂的那种Groupon类型的模式，在Groupon产生之前，在2006年，淘宝的产品经理一灯就提出了“团购”这种产品。一灯最初的设想是让买家在社区发起团购，“团长”找到足够的人之后，

去跟卖家砍价，这类似于现在蘑菇街的“自由团”。但由于种种原因，在开发的时候，对产品的功能做了裁剪，与最初的设想比起来偏离了一点，变成了让卖家设置团购价，在买家达到指定的数量之后，以团购价成交。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，但最终没有支撑下去，这种交易方式最大的弱点就是让买家看到了卖家的底牌，即便达不到团购的数量，他们也往团购的价格上砍。当时为了提高流量，淘宝网开辟了团购专区，实诚的卖家在达不到团购数量的时候，被砍价砍亏了，狡猾的卖家干脆提高原价，利用这个专区做促销。在接下来的两年里，这个产品沦落成了促销工具（话说现在满大街的团购，其实也就是促销）。这个产品让研发人员对“产品”这个概念有了深刻的认识。

再说一个更加悲剧的——“我的淘宝”。“我的淘宝”是给会员管理自己的商品、交易、收货地址、评价、投诉的地方，这个地方必须在登录之后才能看到，所以风格与外观完全不一样，很长时间都没有优化过，样子丑，用户操作也不方便，如果一个人有很多商品，上下架需要一个一个地操作，非常麻烦（想想那些卖书的）。这时候一个重要人物承志（现在的蘑菇街CEO）登场了，他给我们演示了最牛的前端交互技术，就是Gmail上那种AJAX的交互方式，可以拖动，可以用鼠标右键，也可以用组合键，操作完毕还不刷新页面，管理商品如有神助，帅呆了。我是这个项目的项目经理，一灯是产品经理，我们再拉上万剑和一

伙工程师就开始行动了。我们热火朝天地干了三个月，快要完成的时候，老马突然出现在我身后，看我操作了一遍新版“我的淘宝”之后，问我这是不是客户端软件，我说是网页，他抓狂了，说这跟客户端软件一样，链接下面的下画线都没有，上下架用文件夹表示，他都不知道怎么操作，卖家肯定也不会玩。

页面如下图所示：看看这神乎其技的翻页条、精致的文件夹结构、人性化的多选框，还有一个类似Excel的冻结窗口的功能，这有多么性感啊！

老马果然是神一样的人物，他说的应验了，淘宝历史上第一个群体性事件爆发了，试用完新版本的“我的淘宝”之后，很多卖家愤怒了，说不会玩儿。一灯就和承志一起商量怎么把页面改得像网页一点，改了半个月，愤怒依然没有平息。我很无奈地看着这两个人在那里坚持，然后跟老板们商量怎么办。后来我们到论坛上让大家投票要不要使用新版“我的淘宝”，投票结果是一半以上的人反对。于是这十来个人做了3个月的系统被杀掉了。这让我非常沮丧，但最痛苦的还不是这个，我们下线之后，另外一拨卖家不满了，说这么好的功能怎么没有了？这个产品带给我们的是新技术（AJAX、prototype框架）的尝试，以及新技术对用户操作习惯的改变，一定要慎之又慎。另外，还有一点没有总结好的教训，就是应对群体事件的时候，我们手足无措，在后来“招财进宝”和淘宝商城出现群体性事件的时候，我发现悲剧在重演。

说到“招财进宝”，这个是最悲剧的产品。在2006年“五一”的时候，一个划时代的项目启动了。财神说要用最好的项目阵容，我被选中了，这一下让我觉得我能划分到最好的员工之类，在“我的淘宝”这个产品中严重受伤的心又痊愈了。这是一个商品P4P的系统，就是按成交付费。我们认为已经有很多卖家有钱了，但淘宝上这么多的商品，他们很难被找到，卖家愿意花钱让商品排在前面。我们允许卖家购买广告位，把他的商品按一定算法给出排名（类似于百度的竞价排名，但不仅仅看他出了多少钱，还要看信用、成交量、被收藏数量等，这个算法弄得巨复杂）。这是一个多么牛的盈利模式啊！

这个系统进行得很顺利，但发布的时候，更大的群体性事件出来了，买家们质疑：你们不是承诺三年不收费吗？收广告费不是收费吗？后来我们的竞争对手又推波助澜，公关公司和圈子里各路大侠上蹿下跳，甚至同行推出了“一键搬家”的功能来收纳我们的会员。一时间，舆论哗然，各种矛头都指了过来。为了收场，我们又一次在论坛中让用户投票决定产品是否下线，同“我的淘宝”一样，以悲剧收场。也如同“我的淘宝”一样，下线后，一拨尝到甜头的卖家说，这么好的功能怎么没有了？（直到Yahoo中国合并之后，开发了淘宝直通车，才以类似的产品形态满足了这部分需求。）

虽然“招财进宝”失败了，但这个项目中对技术的探索更加深入，其中用到了用户行为追踪、AJAX等，而且有一个技术的细

节非常经典，淘宝商品详情页面每天的流量有几个亿，里面的内容都是放在缓存里的，做“招财进宝”的时候，我们要给卖家显示他的商品被浏览的次数（见下图），这个数字必须实时更新，而用缓存一般都是异步更新的，所以，一开始根本没考虑把这个数据放入缓存里。我们的商品表里增加了这样一个字段，每增加一个PV，该字段就要更新一次。发布一个小时后，数据库就挂掉了。数据库撑不住怎么办？一般的缓存策略是不支持实时更新的，这时候多隆大神想了个办法，在Pache上面写了一个模块，这个数字根本不经过下层的WebApp容器（只经过Apache）就写入一个集中式的缓存区了，这个缓存区的数据再异步更新到数据库。这就是我前面提到的，整个商品详情的页面都在缓存中了，把缓存用到了极致。

接下来，我们就说说缓存的技术吧。

淘宝在很早就开始使用缓存技术了，在2004年的时候，我们使用一个叫做ESI（Edge Side Includes）的缓存（Cache）。在决定采用ESI之前，多隆试用了Java的很多Cache，但都比较重，后来用了Oracle Web Cache，也经常挂掉，Oracle Web Cache也支持

ESI，多隆由此发现了ESI这个好东东。ESI是一种数据缓冲/缓存服务器，它提供将Web网页的部分（这里指页面的片段）进行缓冲/缓存的技术及服务。以往的数据缓冲服务器和信息传送服务以“页”为单位制作，复制到数据缓冲服务器中，这用于处理静态页面很有效，但在面对动态内容的时候，就很难得到高效率。在ESI中是部分的缓冲网页，使用基于XML的标记语言，指定想要缓冲的页面部分。由此，页面内分为动态地变更部分和静态的不变更部分，只将静态的部分有效地发送到服务器中。淘宝网的数据虽然大部分都是动态产生的，但页面中的静态片段也有很多，例如，页面的头、尾，商品详情页面的卖家信息等（如下图右侧），这些最早都是从ESI缓存中读取的。

ESI解决了页面端静态片段的缓存，聪明的读者可能马上就想到了，在后端的那些数据能不能使用缓存？显然也是可以的，而

且是必需的。例如，一个大卖家的商品和店铺，一天的浏览量可能是几百万个，一个小卖家的可能只有几个，那么这个大卖家的用户信息要是每次都从数据库中读取，显然不划算，要是把这个信息放在内存中，每次都从内存里取，性能要好很多。这种应用场景就是memcached这种Key-Value缓存的用武之地。只可惜，在淘宝急需要memcached的时候，它还没有崭露头角（它于2003年6月出现，但近几年才火爆起来，当时没发现它）。我们的架构师多隆大神再一次出手写了一个缓存系统，叫TBstore，这是一个分布式的基于Berkeley DB的缓存系统，推出之后，在阿里巴巴集团内部使用非常广泛，特别是对于淘宝，TBstore上应用了ESI（就是上面说过的那个ESI）、Checkcode（验证码）、Description（前文说过的商品详情）、Story（心情故事，商品信息里面的一个大字段，长度仅次于商品详情）、用户信息等内容。

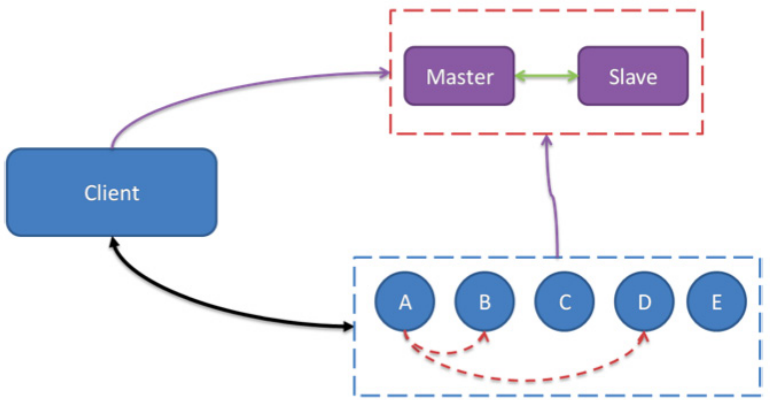
TBstore的分布式算法实现：根据保存的Key（关键字），对key进行Hash算法，取得Hash值，再对Hash值与总Cache服务器数据取模。然后根据取模后的值，找到服务器列表中下标为此值的Cache服务器。由Java Client API封装实现，应用无须关心。

TBstore有一个优点，这也是它的弱点，它的存储是基于Berkeley DB的，而Berkeley DB在数据量超过内存的时候，就要往磁盘上写数据了，所以，它是可以做持久化存储的。但是一旦往磁盘写入数据，作为缓存的性能就大幅下降。

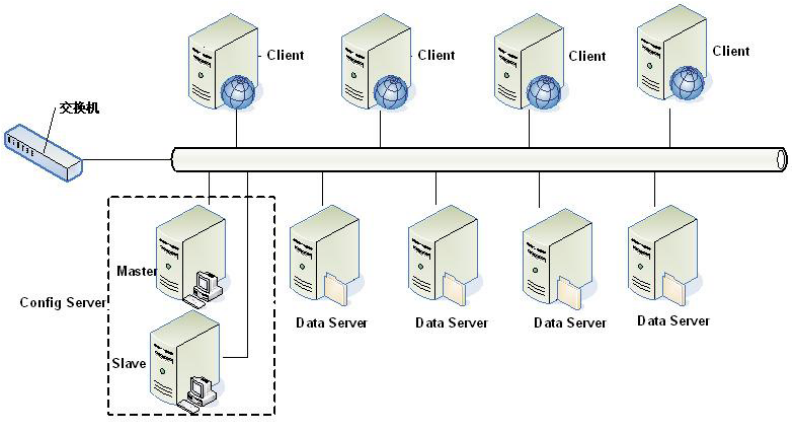
这时又有一个项目，推动了淘宝在缓存方面的技术提升。在2007年，我们把淘宝的用户信息独立出来，形成一个中心系统UIC（User Information Center），因为淘宝所有的功能都要依赖于用户信息，所以这个模块必须单独拿出来，否则以后的系统无法扩展。把UIC拿出来以后，应用系统访问UIC，UIC访问数据库取得用户信息，粗算一下，每天要取几十亿条的用户信息，若直接查询数据库，数据库肯定会崩溃，这里必须要用缓存。于是多隆专门为UIC写了一个缓存系统，取名叫做TDBM。TDBM抛弃了Berkeley DB的持久功能，数据全部存放在内存中。到2009年，多隆又参考了memcached的内存结构，改进了TDBM的集群分布方式，在内存利用率和吞吐量方面又做了大幅提升，推出了TDBM 2.0系统。

下图是一个关键应用的实时监控信息，第一行是UIC的缓存命中率，可以看到有99.2%之高。换句话说，也就是给数据库减少了99.2%的压力。

由于TDBM、TBstore的数据接口和用途都很相似，开发团队把二者合并，推出了淘宝自创的Key-Value缓存系统——Tair（TaoBao Pair的意思，Pair即Key-Value数据对）。Tair包括缓存和持久化两种存储功能。Tair作为一个分布式系统，由一个中心控制节点和一系列的服务节点组成，我们称中心控制节点为Config Server，服务节点是Data Server。Config Server 负责管理所有的Data Server，维护Data Server的状态信息。Data Server 对外提供各种数据服务，并以心跳的形式将自身的状况汇报给Config Server。Config Server是控制点，而且是单点，目前采用一主一备的形式来保证其可靠性。所有的Data Server 地位都是等价的。Tair的架构图如下图所示。

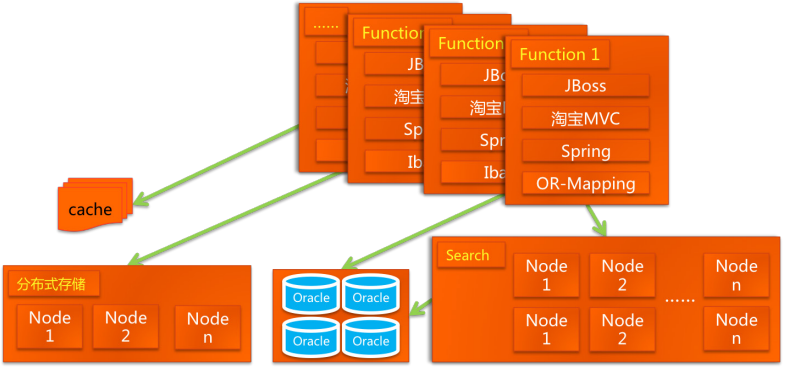


系统部署结构如下图所示。



目前，Tair支撑了淘宝几乎所有系统的缓存信息。Tair已开源，地址为code.taobao.org。

在创造了TFS和Tair之后，整个系统的架构如下图所示。



在这个时候，研发部对搜索引擎iSearch也进行了一次升级，之前的搜索引擎是把数据分到多台机器上，但是每份数据只有一份，现在是每份数据变成多份，整个系统从一个单行的部署变成了矩阵，能够支撑更大的访问量，并且做到很高的可用性。

到2007年，淘宝网日均PV达到2.5亿个，商品数超过1亿个，注册会员数达5千多万个，全网成交额达433亿元。

第三部分

淘宝技术发展3

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，只要把功能实现即可，但作为架构师，要考虑系统的扩展性、重用性，对于这种敏锐的感觉，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，周悦虹开发的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。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，最直观的是页眉和页脚，一开始，我们的每个系统中都复制了一份，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”。后来我就把这部分Velocity模板独立出来做成了公用的模块。

上面说的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，在类目中引入了属性的概念。项目的代号叫做“泰山”，如同它的名字一样，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，且越来越复杂，这样，买家如果要查找一件商品，就要逐级打开类目，找商品之前要弄清商品的分类。而淘宝运营部门管理类目的小二也发现了一个很严重的问题，例如，男装里有T恤、T恤下面有耐克、耐克有纯棉的，女装里也有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、品牌和材质呢，还是先分品牌，再分款式、材质和男女装呢？弄得很乱。这时候，一位大侠出来了——一灯，他说品牌、款式、材质等都可以叫做“属性”，属性是类似Tag（标签）的一个概念，与类目相比更加离散、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外，父节点也可能有属性，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全球最大的，几乎没有什么类目的商品在淘宝上找不到（除了违禁的），但最初的类目属性

改造完之后，我们很缺乏属性数据，尤其是数码类。从哪里弄这些数据呢？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，给运营工作带来了很大的便利，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，都要在类目属性上做调整，让买家更容易找到。例如，夏天让用户在女装一级类目下标出材质是不是蕾丝的、是不是纯棉的，冬天却要把羽绒衣调到女装一级类目下，什么流行，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，随之而来的问题就出现了——调整到哪个类目，所属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，他们肯定受不了。到了2008年，我们研究了超市里前后台商品的分类，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），后台仓库里要按照自然类目来存储，二者密切关联，却又相互分开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摆放商品的类目和属性。改造后的类目属性服务取名为Forest（森林，与类目属性有点神似。Catserver还用于提供卖家授权、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

务量和代码量还是呈爆炸式的增长。业务方总在后面催，开发人员不够就继续招人，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后，就发布上线。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“你写一段代码，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，而业务方还认为开发人员办事不力。

大概是在2007年年底的时候，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，都必须对整个系统回归测试一遍；多个日常修改如果放在一个版本中，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”，即任何一个乘客没有上车，都不许发车。这样做最直接的后果就是火车一直晚点，新功能上线更慢了，我们能明显感觉到业务方的不满，空闻的压力肯定非常大。

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。另外，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，我们叫它UIC（User Information Center）。在UIC中，它只处理最基础的用户信息操作，例如，getUserById、getUserByName等。

在另一方面，还有两个新兴的业务对系统基础功能的拆分也提出了要求。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。但用到的会员功能和交易功能是与主站差不多的，当时做起来就很纠结，因为如果在主站中做，会有一大半跟主站无关的东西，如果重新做一个，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。现在在“我的淘宝”中查看交易记录的时候，还能发现“已买到的宝贝”中把机票和彩票另外列出来了，他们没有加入到普通的订单中。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。

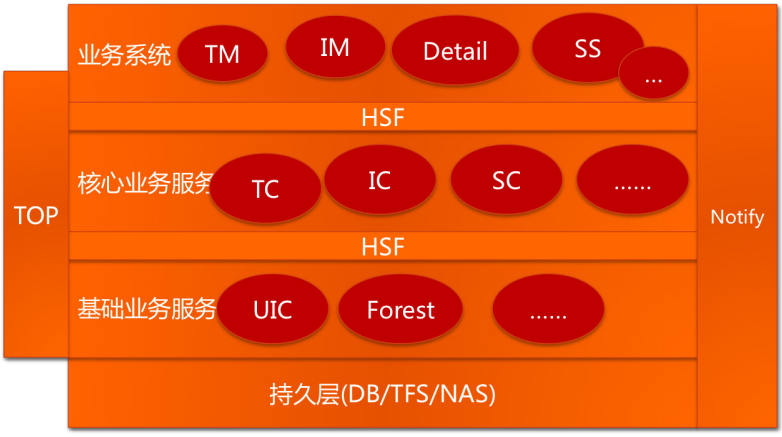
到2008年初，整个主站系统（有了机票、彩票系统之后，把原来的系统叫做主站）的容量已经到了瓶颈，商品数在1亿个以上，PV在2.5亿个以上，会员数超过了5000万个。这时Oracle的连接池数量都不够用了，数据库的容量到了极限，即使上层系统加机器也无法继续扩容，我们只有把底层的基础服务继续拆分，从底层开始扩容，上层才能扩展，这才能容纳以后三五年的增长。

于是我们专门启动了一个更大的项目，即把交易这个核心业务模块拆分出来。原来的淘宝交易除了跟商品管理耦合在一起，还在支付宝和淘宝之间转换，跟支付宝耦合在一起，这会导致系统很复杂，用户体验也很不好。我们把交易的底层业务拆分出

来，叫交易中心（Trade Center，TC），所谓底层业务，就如创建订单、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（Trade Manager，TM），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中完成。这个项目取了一个很没有创意的名字——“千岛湖”，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。这个时候还有一个淘宝商城的项目在做，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。

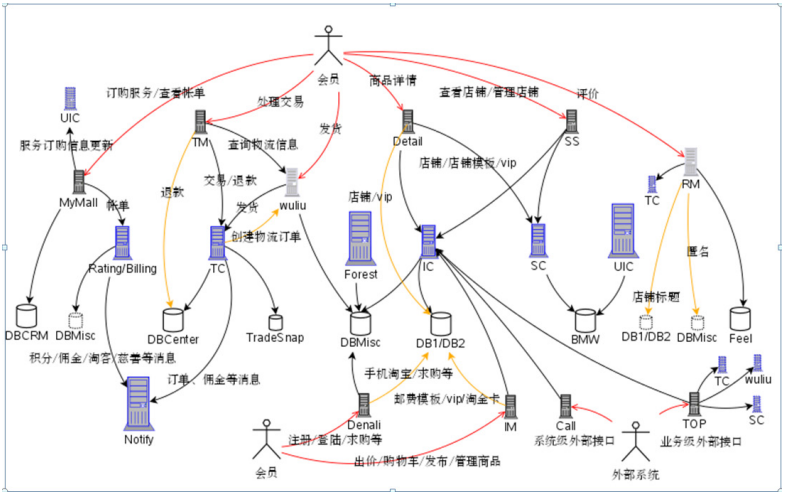
类目属性、用户中心、交易中心，随着这些模块的逐步拆分和服务化改造，我们在系统架构方面也积累了不少经验。到2008年年底就做了一个更大的项目，把淘宝所有的业务都模块化，这是继2004年从LAMP架构到Java架构之后的第二次脱胎换骨。我们对这个项目取了一个很霸气的名字——“五彩石”（女娲炼石补天用的石头）。这个系统重构的工作非常惊险，有人称为“给一架高速飞行的飞机换发动机”。“五彩石”项目发布之后，相关工程师去海南三亚玩了几天。

他们把淘宝的系统拆分成了如下架构。



其中，UIC和Forest在上文已说过，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，后来改名叫SS，即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，示意图如下所示。



系统这么拆分的好处显而易见，拆分之后的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信？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，一个需要解决的问题是用户在A系统登录后，到B系统的时候，用户的登录信息怎么保存？这又涉及一个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？

中间件

互联网系统的发展看似非常专业，其实在生活中也存在类似的“系统”，正如一位哲学家说“太阳底下无新事”。我们可以从生活中的一个小例子来看网站系统的发展，这个例子是HSF的作者毕玄写的。

一家小超市，一个收银员，同时还兼着干点其他的事情，例如，打扫卫生、摆货。

来买东西的人多起来了，排队很长，顾客受不了，于是增加了一个收银台，雇了一个收银员。

忙的时候收银员根本没时间去打扫卫生，超市内有点脏，于是雇了一个专门打扫卫生的。

随着顾客不断增加，超市也经过好几次装修，由以前的一层变成了两层，这个时候所做的事情就是不断增加收银台、收银员和打扫卫生的人。

在超市运转的过程中，老板发现一个现象，有些收银台排很长的队，有些收银台排的人不多，了解后知道是因为收银台太多了，顾客根本看不到现在各个收银台的状况。对于这个现象，一种简单的方法就是继续加收银台。但一方面，超市没地方可加收

银台了，另一方面，作为老板，当然不需要雇太多的人，于是开始研究怎样让顾客了解到收银台的状况，简单地加了一个摄像头和一个大屏幕，在大屏幕上显示目前收银台的状况，这样基本解决了这个问题。

排队长度差不多后，又出现了一个现象，就是有些收银台速度明显比其他的慢，原因是排在这些收银台的顾客买的东西特别多，于是又想了一招，就是设立专门的10件以下的通道，这样买东西比较少的顾客就不用排太长的队了，这一招施展后，顾客的满意度明显提升，销售额也好了不少，后来就继续用这招应对团购状况、VIP 状况。

在解决了上面的一些烦心事后，老板关注到了一个存在已久的现象，就是白天收银台很闲，晚上则很忙，于是从节省成本上考虑，决定实行部分员工只在晚上上班的机制，白天则关闭一些收银台，顾客仍然可以通过大屏幕看到哪些收银台是关闭的，避免走到没人的收银台去，实行这招后，成本大大降低了。

这个生活中的例子及其解决的方法，其实和互联网网站发展过程中的一些技术是非常类似的，只是在技术层面用其他名词来表达了而已，例如，有集群、分工、负载均衡、根据QoS分配资源等。

- 集群：所有的收银员提供的都是收银功能，无论顾客到哪一个收银员面前，都可完成付款，可以认为所有的收银员就构成了一个集群，都希望能做到顾客增加的时候只需增加收银员就行。在现实生活中有场地的限制，而在互联网应用中，能否集群化还受限于应用在水平伸缩上的支撑程度，而集群的规模通常会受限于调度、数据库、机房等。

- 分工：收银员和打扫卫生的人分开，这种分工容易解决，而这种分工在互联网中是一项重要而复杂的技术，没有现实生活中这么简单，涉及的主要有按功能和数据库的不同拆分系统等，如何拆分以及拆分后如何交互是需要面临的两个挑战。因此，会有高性能通信框架、SOA平台、消息中间件、分布式数据层等基础产品的诞生。

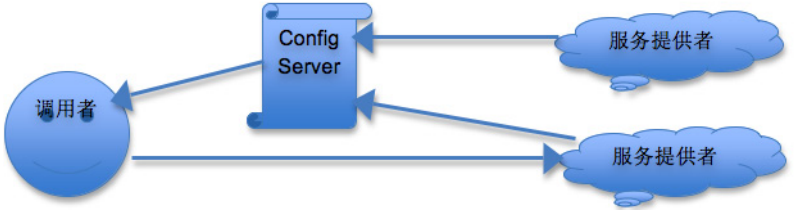
- 负载均衡：让每个收银台排队差不多长，设立小件通道、团购通道、VIP通道等，这些可以认为都是集群带来的负载均衡的问题，从技术层面上说，实现起来自然比生活中复杂很多。

- 根据QoS分配资源：部分员工仅在晚上上班的机制要在现实生活中做到不难，而对互联网应用而言，就是一件复杂而且极具挑战的事。

参照生活中的例子来说，在面对用户增长的情况下，想出这

些招应该不难，不过要掌握以上四点涉及的技术就相当复杂了，而且互联网中涉及的其他很多技术还没在这个例子中展现出来，例如缓存、CDN等优化手段；运转状况监测、功能降级、资源劣化、流控等可用性手段，自建机房、硬件组装等成本控制手段。因此，构建一个互联网网站确实是不容易的，技术含量十足，当然，经营一家超市也不简单。

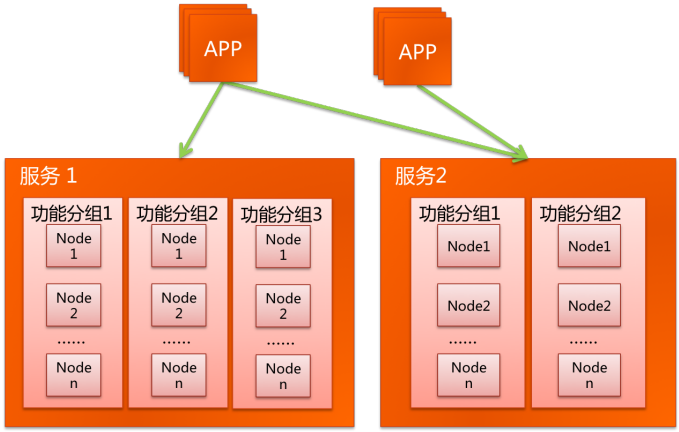
从超市的运维可以抽象出系统设计的一些思路，服务拆分之后，如何取得我需要的服务？在“电视机”上，把每个集群能提供的服务显示出来。你不需要关心哪个人为你服务，当你有需要的时候，请先看头顶的电视机，它告诉你哪个服务在哪个区域。当你直接去这个区域的时候，系统会给你找到一个最快速的服务通道。



这就是HSF的设计思想，服务的提供者启动时通过HSF框架向ConfigServer（类似超市的电视机）注册服务信息（接口、版本、超时时间、序列化方式等），这样ConfigServer上面就定义了所有可供调用的服务（同一个服务也可能有不同的版本）；服

务调用者启动的时候向ConfigServer注册对哪些服务感兴趣（接口、版本），当服务提供者的信息变化时，ConfigServer向相应的感兴趣的服务调用者推送新的服务信息列表；调用者在调用时则根据服务信息的列表直接访问相应的服务提供者，而无须经过ConfigServer。我们注意到ConfigServer并不会把服务提供者的IP地址推送给服务的调用者，HSF框架会根据负载状况来选择具体的服务器，返回结果给调用者，这不仅统一了服务调用的方式，也实现了“软负载均衡”。平时ConfigServer通过和服务提供者的心跳来感应服务提供者的存活状态。

在HSF的支持下，服务集群对调用者来说是“统一”的，服务之间是“隔离”的，这保证了服务的扩展性和应用的统一性。再加上HSF本身能提供的“软负载均衡”，服务层对应用层来说就是一片“私有云”了。

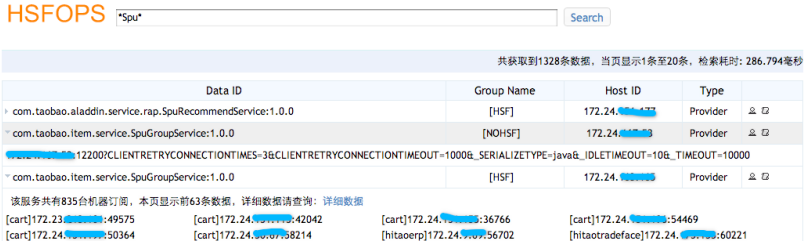


HSF框架以SAR包的方式部署到Jboss、Jetty或Tomcat下，在应用启动的时候，HSF（High-Speed Service Framework，在开发团队内部有一些人称HSF为“好舒服”）服务随之启动。HSF旨在为淘宝的应用提供一个分布式的服务框架，HSF从分布式应用层面以及统一的发布/调用方式层面为大家提供支持，从而可以很容易地开发分布式的应用以及提供或使用公用功能模块，而不用考虑分布式领域中的各种细节技术，例如，远程通讯、性能损耗、调用的透明化、同步/异步调用方式的实现等问题。

从上图HSF的标志来看，它的速度是很快的。HSF是一个分布式的标准Service方式的RPC（Remote Procedure Call Protocol，远程过程调用协议）框架，Service的定义基于OSGI的方式，通讯层采用TCP/IP协议。关于分布式的服务框架的理论基础，HSF的作者毕玄写了一篇博文（http://www.blogjava.net/BlueDavy/archive/2008/01/24/177533.html），有关基于OSGI的分布式服务框架，也有一系列的博文（http://www.blogjava.net/BlueDavy/archive/2008/01/14/175054.html）。

从下面这个HSF监控系统的截图中可以更直观地看到一些信

息，在两个集群中有两个服务器（其实有更多的，没有全部截图下来）都提供com.taobao.item.service.SpuGroupService 这一服务，版本号都是1.0.0，这个服务在ConfigServer上的注册信息中包含超时时间、序列化方式。在后面那条信息中可看到，在展开的这个集群中服务有835台机器已订阅，这些订阅者有淘宝的服务器（cart是购物车功能的服务器），也有hitao（淘花网）的服务器。



HSF系统目前每天承担了300亿次以上的服务调用。

一些读者可能会有一个疑问：既然淘宝的服务化是渐进式的，那么在HSF出现之前，系统之间的调用采用什么方式呢?

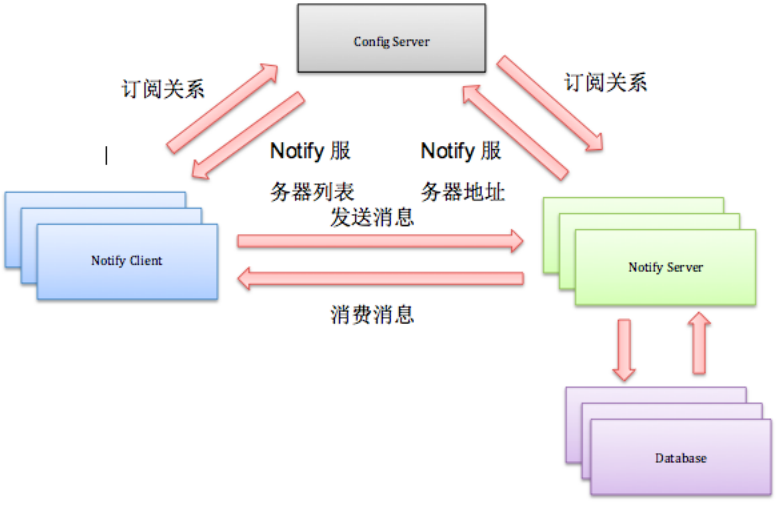
这个有点“五花八门”，例如，对于类目的调用方式是：Forest打包成一个JAR包，在应用启动的时候装载到内存中，仅这一个JAR包所占用的内存就有800MB之多（因为淘宝的类目数据太庞大了），对于当时一般只有2GB内存的开发机来说，加载完类目信息后，机器运行速度就非常慢。对于用户信息（UIC）

来说，一开始的调用方式是用Hessian接口。还有一些系统是通过WebService、Socket甚至是HTTP请求来相互调用的。每种调用方式都涉及各种超时、信息的加解/密、参数的定义等问题，由此可见，在没有HSF之前，系统之间的调用是错综复杂的。而随着系统拆分得越来越多，必须由一个统一的中间层来处理这种问题，HSF正是在这种背景下诞生的。

Notify

HSF解决了服务调用的问题，我们再提出一个很早就说过的问题：用户在银行的网关付钱后，银行需要通知到支付宝，但银行的系统不一定能发出通知；如果通知发出了，不一定能通知到；如果通知到了，不一定不重复通知一遍。这个状况在支付宝持续了很长时间，非常痛苦。支付宝从淘宝剥离出来的时候，淘宝和支付宝之间的通信也面临同样的问题，那是2005年的事情，支付宝的架构师鲁肃提出用MQ（Message Queue）的方式来解决这个问题，我负责淘宝这边读取消息的模块。但我们发现消息数量上来之后，常常造成拥堵，消息的顺序也会出错，在系统挂掉的时候，消息也会丢掉，这样非常不保险。然后鲁肃提出做一个系统框架上的解决方案，把要发出的通知存放到数据库中，如果实时发送失败，再用一个时间程序来周期性地发送这些通知，系统记录下消息的中间状态和时间戳，这样保证消息一定能发出，也一定能通知到，且通知带有时间顺序，这些通知甚至可以实现事务性的操作。

在“千岛湖”项目和“五彩石”项目之后，淘宝自家的系统也拆成了很多个，他们之间也需要类似的通知。例如，拍下一件商品，在交易管理系统中完成时，它需要通知商品管理系统减少库存，通知旺旺服务系统发送旺旺提醒，通知物流系统上门取货，通知SNS系统分享订单，通知公安局的系统这是骗子……用户的一次请求，在底层系统可能产生10次的消息通知。这一大堆的通知信息是异步调用的（如果同步，系统耦合在一起就达不到拆分的目的），这些消息通知需要一个强大的系统提供支持，从消息的数量级上看，比支付宝和淘宝之间的消息量又上了一个层次，于是按照类似的思路，一个更加强大的消息中间件系统就诞生了，它的名字叫做Notify。Notify是一个分布式的消息中间件系统，支持消息的订阅、发送和消费，其架构图如下所示。

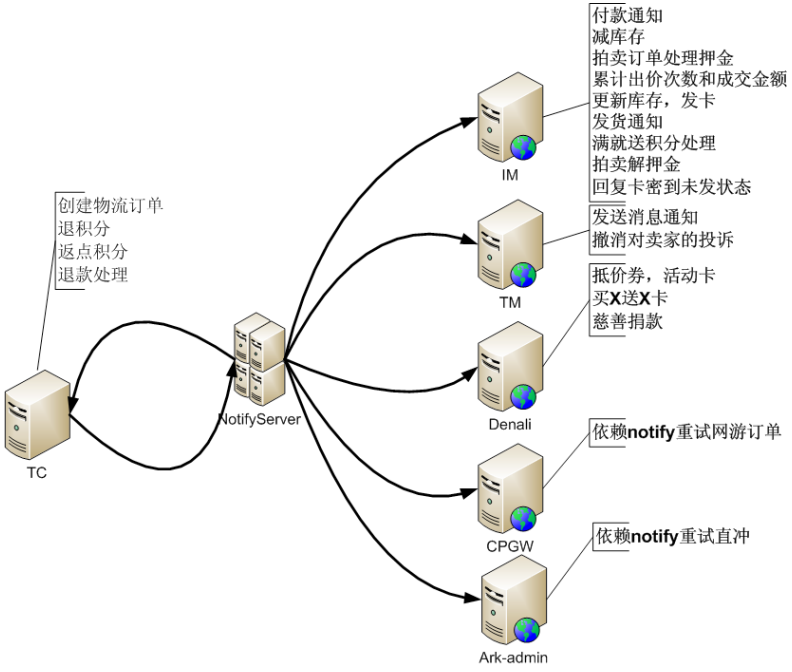


NotifyServer在ConfigServer上面注册消息服务，消息的客户端通过ConfigServer订阅消息服务。某个客户端调用NotifyServer发送一条消息，NotifyServer负责把消息发送到所有订阅这个消息的客户端（这个过程参照HSF一节，原理是一样的）。为了保证消息一定能发出，且对方也一定能收到，消息数据本身就需要记录下来，这些信息存放在数据库中（可以是各种数据库）。由于消息具有中间状态（已发送、未发送等），应用系统通过Notify可以实现分布式事物——BASE（基本可用（Basically Available）、软状态 （Soft State）、最终一致（Eventually Consistent））。NotifyServer可以水平扩展，NotifyClient也可以水平扩展，数据库也可以水平扩展，从理论上讲，这个消息系统的吞吐量是没有上限的，现在Notify系统每天承载了淘宝10亿次以上的消息通知。

下图展示了创建一笔交易之后，TC（交易中心）向Notify发送一条消息，后续Notify所完成的一系列消息通知。

TDDL

有了HSF和Notify的支持，在应用级别中，整个淘宝网的系统可以拆分了，还有一个制约系统规模的更重要的因素，就是数据库，也必须拆分。



在第二部分中讲过，淘宝很早就对数据进行过分库的处理，上层系统连接多个数据库，中间有一个叫做DBRoute的路由来对数据进行统一访问。DBRoute对数据进行多库的操作、数据的整合，让上层系统像操作一个数据库一样操作多个库。但是随着数据量的增长，对于库表的分法有了更高的要求，例如，你的商品数据到了百亿级别的时候，任何一个库都无法存放了，于是分成2个、4个、8个、16个、32个……直到1024个、2048个。好，分成这么多，数据能够存放了，那怎么查询它？这时候，数据查询的中间件就要能够承担这个重任了，它对上层来说，必须像查询一

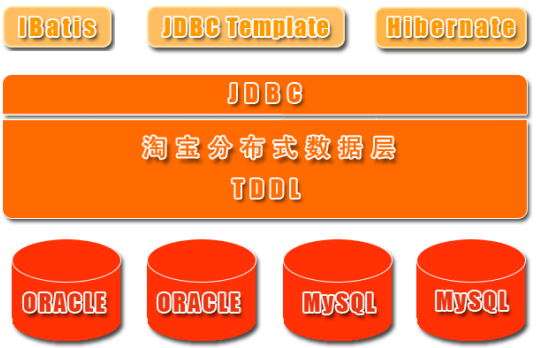
个数据库一样来查询数据，还要像查询一个数据库一样快（每条查询在几毫秒内完成），TDDL就承担了这样一个工作。

另外，加上数据的备份、复制、主备切换等功能，这一套系统都在TDDL中完成。在外面有些系统也用DAL（数据访问层）这个概念来命名这个中间件。

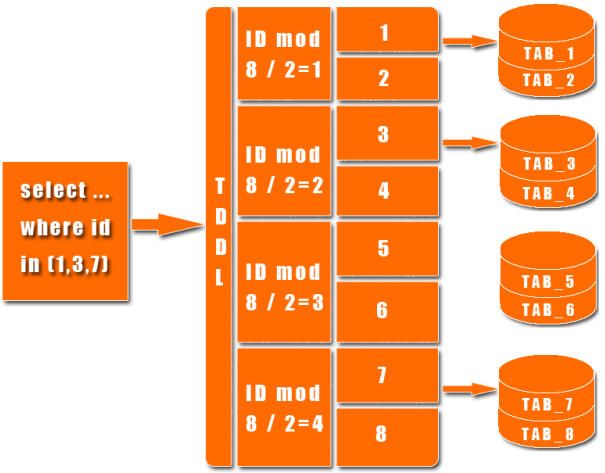
TDDL实现了下面三个主要的特性：

- 数据访问路由——将针对数据的读写请求发送到最合适的地方；
- 数据的多向非对称复制——一次写入，多点读取；
- 数据存储的自由扩展——不再受限于单台机器的容量瓶颈与速度瓶颈，平滑迁移。

下图展示了TDDL所处的位置。



下图展示了一个简单的分库分表数据查询策略。



下面是TDDL的主要开发者之一沈询描述的“TDDL的前世今生”——数据层的发展历程。

CommonDAO的时代

数据切分并不算是一个很新的概念，当商品库切分为两个时，就已经出现了名字叫做xingdian（笑，那时候行癫已经不写代码了，但从代码的版本信息可以看到作者）的人写的CommonDAO。

CommonDAO的思路非常简单实用，因为淘宝主要在使用ibatis作为访问数据库的DAO层，所以，CommonDAO的作用就是对ibatis层做了一个很浅的封装，允许你通过商品字串ID的第一个

字符来访问两台数据库中的一台。

比如，如果字符串ID的第一个字符是0~7，那么走到数据库1去，如果是8~f，则走到数据库2去。同时，也允许用户直接给定数据库的名字来访问数据库。

这应该是最早的数据层原型。

TDDL 1.0时代

后来，大家逐渐发现，如果按照业务的发展规模和速度，那么使用高端存储和小型机的Oracle存储的成本将难以控制，于是降低成本就成了必然。

如何能够在不影响业务正常发展的前提下，从一定程度上解决成本的问题呢？

“对一部分数据库使用MySQL”，DBA们的决策是这样，于是，分布式数据层的重担就落到了华黎的头上。

别看现在数据水平切分似乎已经成了基础知识。在2007年、2008年，如何设计它还真是让我们伤透了脑筋。

当时的我们，只知道eBay有一个数据层，却不知道如何设计和实现？

于是邀请了当时所有的业务负责人来畅想数据层的样子……

得到了以下需求：

- 对外统一一切数据访问；
- 支持缓存、文件存储系统；
- 能够在Oracle和MySQL之间自由切换；
- 支持搜索引擎。

然后，我们自己的问题与现在大家所问的问题也是完全一样的。

如何实现分布式Join（连接）？——在跨节点以后，简单的Join会变成M×N台机器的合并，这个代价比原来的基于数据库的单机Join大太多了。

如何实现高速多维度查询？——就像SNS中的消息系统，A发给B一个消息，那么A要看到的是我发给所有人的消息，而B要看到的是所有人发给我的消息。这种多维度查询，如何能够做到高效快捷呢？

如何实现分布式事务？——原始单机数据库中存在着大量的事务操作，在分布式以后，分布式事务的代价远远大于单机事务，那么这个矛盾也变得非常明显。

华黎带着我和念冰，坐在那里讨论了一个半月，还是没想出

来……于是决定先动起手来。名字是我起的——Taobao Distributed Data layer（TDDL，后来有人对它取了个外号：“头都大了”⊙﹏⊙b）

学习开源的Amoeba Proxy。

找到的目标应用是“收藏夹”，首先要做的两个关键的特性是：分库分表和异构数据库的数据复制。

开始本来希望和B2B的团队合作，因为我们觉得独立的Proxy没有太大必要。而SQL解析器因为有淘宝特殊的需求，所以也需要重写。

可惜，最后因为B2B的人搬到滨江去了，交流十分不畅，所以最后只是做了拿来主义，没有对开源的Amoeba和当时的Cobar有所贡献。

回到淘宝，因为有东西可以借鉴，我们在一个多月的时间内就完成了TDDL 1.0版本的工作。上线过程中虽然出了点小问题，不过总体来说是比较成功的。

TDDL 2.0时代

随着使用TDDL的业务越来越多，对业务方来说，DBA 对于使用MySQL以及数据切分也积累了比较多的经验，于是决定开始动核心应用了。

“评价”是第一个重要的应用，评价最重要的问题还是在于双向查询、评价、被评价。于是我们的异构数据源增量复制就派上了用场。

然后是“商品”，我们在商品上投入了近半年的时间，失败很多，也成长得最快。

- 容量规划做得不到位，机器到位后因压力过大，直接死掉，于是产生了数据库容量线上压力模拟测试。
- 历史遗留问题，商品几乎是所有的业务都会使用的资源，所以接口设计比较复杂。很多接口的调用在新架构上很难以低成本的方式实现。而推动业务改动，则需要大量的时间和成本。
- 数据层代码被业务代码侵染，看起来似乎应该是数据层的代码，但实际上又只有商品在使用。这种问题让数据层的依赖变得更加庞大，边缘代码变得更多，冲突更明显。

TDDL 3.0～TDDL 4.0时代

在商品之后，似乎所有的应用都可以使用类似的方式来解决业务增长上量的问题。但正当我们志得意满的时候，却被“交易”撞了一个满怀。

我一直很感谢交易线的所有同仁，他们是淘宝草根精神的典

型代表——功能可以做得不那么“漂亮”，但必须减少中间环节，真正做到了实用、干净、简洁。我们在向他们介绍产品的时候，他们对我们的实现细节提出了非常多的质疑，他们认为整个流程中只有规则、主备切换对他们是有意义的，而解析、合并则是他们所不需要的功能。

“不需要的功能为什么要放到流程里？增加的复杂度会导致更多的问题”。在当时，我感到很痛苦，因为我无法回答他们这些质疑之声。

不过，也正是因为这些质疑，让我有了一个契机，重新审视自己所创造出来的产品。

我问自己：它能够给业务带来什么益处？

对此，我的回答是：

- 规则引擎/切分规则可以用配置的方式帮助业务隔离具体的数据库地址与用户的业务逻辑；
- 单机主备切换；
- 数据源简化和管理。

于是，我们就产生了TDDL 3.0版本。其主要的作用就是将代码做了逻辑切分，将单机主备切换和数据源管理独立了出来。这

样，可以针对不同的业务需求，给予不同的逻辑分层。让每一个业务都有适合自己的使用数据库的方式。

同时，我们开始做工具，Rtools/JADE 作为数据库运维平台的组件被提了出来。在它的帮助下，我们发现能够极大地提升用户在使用单机数据源和多机数据源时的效率。用户只需要在客户端给定两个属性，就可以立刻开始使用。结果是用户反馈比以前好了很多。

这也坚定了我们开发工具的决心。

工具平台时代

在尝到工具平台的甜头以后，我们在工具组件上走得更远了。

首先被提出的是“愚公”数据迁移平台。该平台能够在多种异构的数据库中进行数据的平滑移动，对业务影响很小，并且也允许业务插入自己的业务逻辑。

这个东西主要能够帮助业务进行数据库自动扩容，自动缩容，单机、多机数据迁移，在Oracle到MySQL数据迁移等场景中都发挥了重要的作用。

然后，又以内部开源的方式提出了“精卫”数据增量复制平台。这个平台基于数据库的通用数据分发组件，基于开源的

Tungsten进行了大量Bug Fix和结构调优。在数据的一对多分发以及异步通知给DW和搜索等场景中都发挥了重要的作用。

TDDL的现在

粗略统计下来，TDDL已经走过了4年的时间，满足了近700个业务应用的使用需求。其中有交易商品评价用户等核心数据，也有不那么有名的中小型应用。量变产生质变，如何能够更好地帮助这些业务以更低的成本更快地完成业务需求，将成为数据层未来最重要的挑战。

Session框架

介绍Session框架之前，有必要先了解一下Session。Session在网络应用中称为“会话”，借助它可提供客户端与服务系统之间必要的交互。因为HTTP协议本身是无状态的，所以经常需要通过Session来解决服务端和浏览器的保持状态的解决方案。用户向服务器发送第一个请求时，服务器为其建立一个Session，并为此Session创建一个标识，用户随后的所有请求都应包括这个标识号。服务器会校对这个标识号以判断请求属于哪个Session。会话保持有效，默认状况下，直到浏览器关闭，会话才结束。

Session中存储的内容包括用户信息：昵称、用户ID、登录状态等。

当网站服务器只有一台的时候，用Session来解决用户识别是很简单的，但是当网站是一个集群的时候，同一用户的两次请求可能被分配到两台不同的服务器上处理。怎样保证两次请求中存取的Session值一致呢？还有一个问题：网站规模扩大时，对于一个具有上亿个访问用户的系统来说，当大部分用户的Session信息都存储在服务端时，要在服务端检索出用户的信息效率就非常低了，Session管理器不管用什么数据结构和算法都要耗费大量内存和CPU时间。如何解决服务端Session信息的管理？

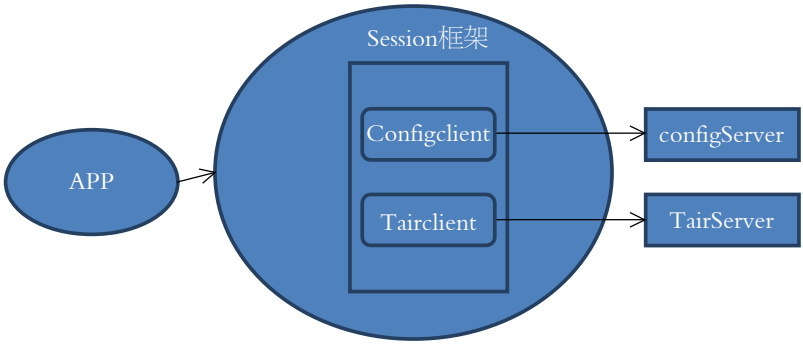
解决集群Session共享的问题，通常有以下两种办法。

- 硬件负载，将用户请求分发到特定的服务器。
- Session复制，就是将用户的Session复制到集群内所有的服务器。

这两种方法的弊端也很明显：

- 成本较高。
- 性能差。当访问量增大的时候，带宽增大，而且随着机器数量的增加，网络负担成指数级上升，不具备高度可扩展性，性能随着服务器数量的增加急剧下降，而且容易引起广播风暴。

这种情况下，Tbsession框架闪亮登场了。Tbsession框架致力于解决以下几个问题。



- Session的客户端存储，将Session信息存储到客户端浏览器Cookie中。
- 实现服务端存储，减少Cookie使用，增强用户信息的安全性，避免浏览器对Cookie数量和大小的限制。
- Session配置统一管理起来，集中管理服务端Session和客户端Cookie的使用情况，对Cookie的使用做有效的监管。
- 支持动态更新，Session的配置动态更新。

简单地说，就是要么用客户端Cookie来解决问题，要不用服务端的集中缓存区（Tair）的Session来解决登录问题。Tair已在前文介绍过，Session对它的使用就不再描述了。

为什么这里还要提到用Cookie这种比较“落伍”的方式呢？其实是因为在淘宝3.0版本以前，我们一直都用Cookie来识别用户，Cookie是放在客户端的，每一次HTTP请求都要提交到服务端，在访问量比较小的时候，采用Cookie避免了Session复制、硬件负载等高成本的情况。但随着用户访问规模的提高，我们可以折算一下，一个Cookie大概是2KB的数据，也就是说，一次请求要提交到服务器的数据是网页请求数据，再加上2KB的Cookie，当有上亿个请求的时候，Cookie所带来的流量已经非常可观了，而且网络流量的成本也越来越高。于是在3.0版本中，我们采用了集中式的缓存区的Session方式。

到此为止，应用服务切分了（TM、IM）、核心服务切分了（TC、IC）、基础服务切分了（UIC、Forest）、数据存储切分了（DB、TFS、Tair），通过高性能服务框架（HSF）、分布式数据层（TDDL）、消息中间件（Notify）和Session框架支持了这些切分。一个美好的时代到来了，高度稳定、可扩展、低成本、快速迭代、产品化管理，淘宝的3.0系统走上了历史的舞台。

在这个分布式系统的支持下，更多的业务迅速开发出来了，因为任何一个业务都基于淘宝的商品、交易、会员、评价等基础体系，而这些基础体系就像“云”一样存在，现在可以随处调用了。Hitao、淘花网、良无限、天猫、一淘、聚划算、各种SNS、各种移动客户端等如雨后春笋般地成长起来了。目前，淘宝已经

变成了一个生态体系，包含C2C、B2C、导购、团购、社区等各种电子商务相关的业务。

既然说是一种“生态体系”，那就不能把所有的业务把控在自己的手中，在开发3.0版本的过程中，我们就有一个团队把淘宝“开放”出去了，我们把自己的数据、自己的应用通过接口的方式让更多的开发者调用，他们可以开发出形形色色的产品，例如，你可以开发出心形的淘宝店铺、菱形的店铺，再放到淘宝上供商家购买。淘宝再多的员工，其创造力也是有限的，而开放出去之后，让无限的人都可以参与到这个生态体系的建设中来，这个生态体系才是完整的。下面就是开放平台的架构师放翁所记述的“开放平台这几年”[注1]。

开放平台

2006年年底：阿里巴巴提出了Workatalibaba的战略，二十多个人就被拉到湖畔花园马云的公寓里开始一个叫阿里软件的公司创业。当时对于Work at alibaba有一个朦朦胧胧的感觉，就是要为中小企业提供一个工作平台，但是工作平台又需要是一个开放的平台，因为卖家的需求是长尾的，当时火热的Salesforce给了阿里人

注1　作者放翁在文中涉及的所有技术点都可以在http://blog.csdn.net/cenwenchu79中找到详细的内容，同时本文主要介绍开放平台技术发展历程，产品和业务内容不涵盖在此，因此，受众群体主要是技术人员。——作者注

一些启示，那就是做一个支持二次开发的工作平台，半开放式地满足各种卖家的长尾管理需求。此时，软件市场上就开始培养起最早的一批TP（淘宝开放合作伙伴）。迄今为止，很多非常成功的TP就是从那个时候开始进入淘宝卖家市场的。

但经过一年的平台建设，发现开发者非常难利用平台做二次开发，只有阿里软件公司内部的团队构建了三个不同的CRM软件。这时候淘宝来了一个业界的技术牛人王文彬（花名：菲青），这位淘宝新晋的首席架构师找到阿里软件的平台架构团队，谈到了当时业界还非常新颖的一种技术平台——开放平台。由于阿里软件已经在做类似的开放工作，希望能够以合作的方式来试水开放平台。当时双方都是一种尝试的态度，因此，最后敲定投入一个人花两周时间，看是否能够做出原型，如果可以，就继续做，如果出不了原型，就此结束。两周时间里，负责阿里软件的架构师放翁参看着美国雅虎的开放模式就搞出了开放平台的第一个雏形，没想到就这样开启了5年的开放之路。后面会根据时间轴来说一下开放平台的产品和技术的变革，每一年会发生很多事情，但是调出的一点一滴是当年最有感触的。

2007年：萌芽。SOA盛行的年代，内部架构服务化成为开放的第一步，内部服务不做好隔离，开放就意味着风险不可控。支付宝今天的服务框架SOFA（类ESB）、淘宝的HSF（OSGI）、阿里软件的ASF（SCA）都是那个年代的产物，但服务化带来的痛却是一样的，不论是OSGI还是SCA之类的服务框架，本身的服务

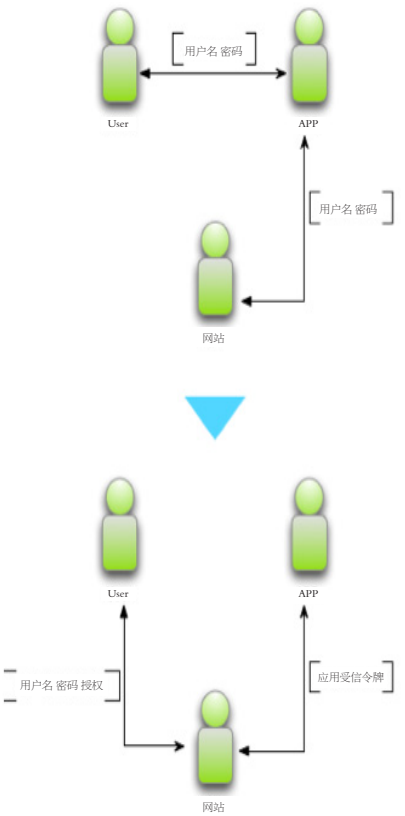
化规约设计都类似，但难题也都摆在每个架构师和开发者面前：服务单元Bundle的粒度控制，服务之间依赖管理，性能与规范的冲突，调试与隔离的平衡。这些都使得一线开发者和平台框架实现者出现非常多的矛盾，而这个过程能活下来的框架最后都是摒弃了很多企业级的设计思路，因为SOA架构从企业级产品演变而来，而服务化后的内部平台要面对的开放平台天生就是互联网的产物。

2008年：雏形。2008年年底，平台开放淘宝服务30个，每天调用量2000次，这一年开放平台的开发者面向的客户主要是阿里巴巴上的中小企业和淘宝C店卖家。开放平台建设初期要解决的就是三个问题：

- 服务路由。（外部可以获取内部信息）
- 服务接口标准化。（统一方式的获得各种标准化信息）
- 授权。（外部合法的获取内部信息）

服务路由其实就是写一个高效的HttpAgent，服务接口标准化就是对象文本化（JSON，XML）。今天在各大开放平台广为使用的OAuth协议，当前处于0.6版本，没有任何实际的互联网开放平台使用，直到Google于2008年年底慢慢地对外推广开放的时候，OAuth被封装到Google的Open SDK中，才使得很多中小型互联网公司使用这种看似复杂的两阶段授权交互模式。淘宝初期采

用的是自有协议，因为OAuth2以前的逻辑较复杂且使用不方便，直到2011年才开始支持OAuth2，同时做了部分的安全增强。授权解决了开放最大的一个问题：用户安全的对应用访问其数据受信。用户从此不用赤裸裸地将用户名和密码交给一个应用软件，应用也可以在允许的范围内（操作、数据、授权时长）充分利用用户授权来玩转创意。



有了上面的三板斧（路由、数据规范和授权），开放平台正式开门迎客了，没有对外做任何的推广，日均调用数据就猛增到了1000次，此时两个互联网的新兴技术Memcached和Hadoop开始在开放平台中尝试。今天看来，这两个技术已经被大规模地使用，Memcached无疑是最好的选择，但当时号称分布式缓存的Memcached其实是集中式缓存的一种，真正的分布式缓存还都在纠结于一致性和效率的问题（第2、3阶段提交）。此时需要有一种方式能够保证效率（可扩展）和稳定性，于是我们封装了Memcached客户端，提升当时BIO的Java客户端的性能，同时引入了客户端负载均衡和容灾的设计，这种设计已经被应用在现在很多大型分布式系统中。另一方面，每天上千万的访问也让技术和产品对访问的行为有很强的分析需求，此时，Hadoop在雅虎的充分利用引起了我们的重视（当时的雅虎技术创新一直都是业界的领头人），通过仅有的两台机器和一堆技术文档，我们摸索着搭建了公司内部的第一个Hadoop集群，而所写的Hadoop入门实践也成为当时Hadoop入门的基础文档，对于每天2000次调用量的日志分析需求来说，Hadoop用得游刃有余，但随着业务的不断发展，Hadoop离线分析所带来的问题也凸显出来，MR程序面对灵活多变的分析需求，显得不易维护且效率低下（数据反复读取分析），于是我们也开始思考怎样改进这个新玩意儿。

2009年：产品化。到2009年年底，平台开放淘宝服务100多个，每天调用量为4000次，这一年开放平台的开发者面对的主要

是淘宝C店卖家，卖家工具成为服务市场的主流。这一年也是变化的一年，阿里软件年中的分拆使得开放平台的归属有些微妙，一种情况是留在阿里云，作为集团的基础设施，另一种情况就是跟着主要的业务需求方淘宝走，最后我们还是说服了博士，结束了阿里软件的老平台，淘宝正式开始自己的开放之路。来到淘宝后，业务开放迅猛增长，从30个API猛增到了100个API，没有对外做任何业务推广，平台调用量到了年底翻番。此时技术上的挑战又聚焦到了性能上，一次API Call的业务消耗平均在30～40ms，开放平台当时的平台处理消耗平均在10ms左右。我们做了数据打点和分析，发现最大的消耗在于互联网数据的接收，同时大量的图片数据上行，更是加大了平台处理时间。另外，从访问日志分析中可以看到很多无效的请求也占用了非常多的处理时间，这也意味着无效请求和有效请求一样在消耗着有限的容器线程资源。于是我们开始尝试自己封装字节流解析模块，按需解析上行数据，一来可以提升数据分析的性能（并行业务和数据增量分析操作），二来可以用最小代价处理异常请求（当发现不满足业务规范时，则立刻丢弃后续所有的数据），这块实现被叫做LazyParser，主要的实现重点就是最小化数据缓存来进行并行业务和数据解析操作，上线后效果不错，整个系统平均处理时间从10ms降低到了4ms。（包含了异常处理的优化和解析性能的提升）

另一方面，Hadoop的MR问题也日益突出，一大堆MR的

Class（类）维护成本高、性能问题也随之出现。此时我们开始尝试抽象分析业务场景，想到的是是否能够通过配置就可以完成各种统计分析需求。要用配置替代代码，其实就看是否可以穷举代码所实现的各种统计需求。当回顾SQL的理念时，发现其实所有的统计在切割成为KV作为输入/输出时，所涵盖的需求无非是Max、Min、Average、Sum、Count、Distinct（这个是2012年实现的，用了bloomfilter和AtomicLong)，再复杂一些无非就是上述几个操作结果的数学表达式运算。因此，KV输入和KV输出的离散统计配置需求已经抽象出来了，接着就是把统计完的一组组KV根据K来做Groupby（分组），就生成了传统意义上的报表（K，v1,v2…）。从此以后，每天的统计需求都通过配置来改变，再也没有一大堆MR代码，同时一次数据输入就可以完成所有分析的处理，性能上得到了极大的提高。

虽然Hadoop每日分析抽象出模型配置解决了性能和易用性的问题，但是对于即时分析却不太适合，当时出于监控的需求，希望能够一个小时就可以对数据做一次增量的分析，用于监控服务整体的调用情况，保证对异常问题的即时排查。由于一天4000次的调用量还不算很大，因此，当时就直接考虑采用MySQL分库分表的方式，然后定时做SQL的查询，结果发现效果不错。当然，这个过程又产生了一个小组件，要直到4000次的日志数据写磁盘和DB双份必然会带来不少的I/O消耗，同时这个系统并不是账务系统，丢掉一点日志也没关系。因此，就采取了异步批量数据外

写的设计（多线程守护各自的一块Buffer页，定时外刷或者满页外刷），这样在双写的情况下，单机的Load也没有超过0.7。

但快到年底的时候，发生了一件事情让我们头痛不已，同时也成了开放平台的一个“隐形炸弹”。一天晚上，突然发生平台大规模拒绝服务的告警，观察整个集群发现，业务处理时间从平均的30～40ms，上升到了1s，仔细观察发现，某一个业务的响应时间大幅攀升，从原来20ms的响应时间飙升到了1s以上，此时由于HTTP请求的同步性，导致前端服务路由网关的集群线程都释放得非常慢，阻塞处理这个业务的请求，而其他正常的业务（淘宝开放平台背后的服务由不同的团队维护，处理时间从1ms到200ms都有）也无法被访问，因此，才有了开始的全线告警的产生。后来发现是这个业务团队的一次发布中忽略了数据库索引建立导致服务耗时增加，但这个问题开始时不时地拜访开放平台，开放平台稳定性受制于任何一个业务方，这是不可接受的。对于这个问题，起先考虑集群拆分，即将重要业务和不重要业务拆分，但考虑到实施成本（不同服务的利用率差异很大）和业务隔离是否彻底（重点业务也会相互影响），最终放弃了这个想法。当时又想到了软负载切割Haproxy和LVS，一个是七层的网络软负载切割，一个是四层的负载切割，由于涉及业务，于是考虑用七层的软负载切割，尝试一台Haproxy挂7台虚拟机，然后运行期可动态调整，配置在出现问题的时候可人工干预切割流量。就这样，我们有了告警以后可以手动切割的半人工方式干预措施。

但我们依然晚上睡不踏实……（期间考虑过Web请求异步化和Servlet3的模式来规避同步HTTP带来的平台阻塞，但当时唯一支持Servlet3的Jetty和Tomcat做压力测试，效果都很不稳定）

2010年：平台化。到2010年年底，平台开放淘宝服务300多个，每天调用量为8亿次，这一年淘宝正式开始对外宣传开放，淘宝开放年赢在淘宝，目前很多年收入上千万的TP在这个时候成了先锋（2010年以前的可以叫做先烈），产品层面上，这一年除了卖家工具的继续发展，SNS热潮的兴起带动了淘江湖的买家应用，游戏应用的淘金者蜂蛹而入，开放的服务也继续保持300%的增速，覆盖面从卖家类延伸到了买家类，从简单的API提供，到了淘宝网站支持深度集成应用到店铺和社区。

在8亿次访问量的情况下，再用MySQL做流式分析已经不可能了，分析时间要求也从一个小时提升到了20分钟，此时经过快一年半的Hadoop使用和学习，再加上对分布式系统的了解，正式开始写第一版的流式分析系统，MR的抽象依旧保留，而底层的数据计算分析改用其他方式，这个“其他方式”和Hadoop的差异在于：

- 分析任务数据来源于远端服务器日志（主要通过Pull，而非Push）。
- 任务分配和调度采用被动分配（有点类似于Volunteer Computing的模式），Mater轻量的管理任务，Slave加入即

可要求执行任务，对任务执行的情况不监控，只简单通过超时来重置任务状态。

- 任务统一由Master来做最后的Reduce，Slave可以支持做Shuffle来减少数据传输量和Master的合并压力，Master负责统一输出结果到本地。

总的来说，就是数据来源变了，数据不通过磁盘文件来做节点计算交互（只在内存使用一次就丢掉了），简化任务调度，简化数据归并。这样第一版本的流式分析出来了，当然，后面这些设计遇到的挑战让这个项目不断在演进，演进的各种优化几年后发现都在Hadoop或者Hive之类的设计中有类似的做法。（参看Blog，地址为http://blog.csdn.net/cenwenchu79，根据时间轴可以看到各种结构优化和性能优化的过程）这个系统三台虚拟机撑住了8亿次的日志即时分析，MySQL日志分析就此结束。

这一年另一个重大改变就是更多的人对开放的价值有所认同，淘宝从一个部门的开放走到了淘宝公司的开放，什么叫做部门开放？就是在10年以前大部分的API开放都是开放平台这个团队来做封装维护，30个API还可以支撑，100个API已经让一个专业的小团队应接不暇（当然不得不承认，迄今为止，淘宝最有全局业务知识的还属这个团队的成员），300多个API这种势头基本上就无法由一个团队来做了，业务变更带来的接口不稳定经常被投诉。因此，我们启动了服务轻量化的“长征项目”，逐渐通过工

具和平台将服务接入变成自动化的方式，将原来开放一个服务需要点对点，手把手花一周时间实施完成的过程，通过自动化服务发布平台，一个人一天时间就可以发布一个服务，并且服务的文档中，多语言版本SDK都自动生成。这样就具备了服务轻量化的基础，然后将各个新开放的业务采用这种模式接入，而老业务逐渐归还给各个业务方去维护。这样，服务的“稳定性”（业务方面）得到了非常大的提升，用户对于服务的满意度也得到了极大的提高。

但这个担子放下了，那个担子又挑上了，在上面谈到后台应用不稳定导致平台整体不稳定的问题在轻量化以后出现的频率和次数更多了，因为发布和维护都落到了后台部门，此时对于各个系统的把控就更弱了，KPI中的稳定性指标基本就没法定了。唯一能够彻底解决问题的办法就是HTTP服务异步化+事件驱动+虚拟隔离线程池。2010年年中对Jetty7做了一次压测，发现Continuations的效果已经可以上正式的环境了，于是开始在Jetty7的基础上做HTTP服务异步化+事件驱动的封装，同时也实现了一个虚拟隔离线程池做配合。具体设计细节这里就不再多说，参看Blog，简单描述原理如下：

- 将前端容器线程和业务处理隔离。（类似NIO和BIO的设计差异）
- 业务处理如果依赖于外部系统，则采用事件驱动的方式来

减少线程等待，同时提高线程占用资源的利用率。（从这点上说，理想和现实还是有很多细节差异的，在实现的时候必须根据依赖系统消耗时间占总时间的比例看是否需要事件驱动，事件驱动带来的切换消耗是比较大的）

- 通过一个大的线程池虚拟设置不同的业务可消耗的最大资源数，来充分共享资源在异常情况下限制业务占用过多的资源（任务处理开始排队，而非无度地占用资源）。

这个组件上线以后，没过几天就发生了一个典型的案例，一个业务在下午2点开始响应时间从10ms上升到了40ms，然后继续上升到200ms，当时给这个业务模拟设置最大的线程资源数是20个，就发现那时候由于RT时间提升，线程资源释放得慢，20个慢慢地被消耗完了，此时这个业务的队列开始从0到100，再到200……（当然，防止内存过多地被占用，会丢弃超过队列长度的业务处理），而其他业务还是正常地使用着资源，平台平稳，到了下午4点多，业务方收到告警修复以后，RT时间下降到了10ms，队列中的请求数量开始减少，最后队列清空，线程资源占用下降到正常水平。

从此以后，震子开心地和我说：开放平台稳定性的KPI可以随便大胆地写几个9了。

2011年：市场化。到2011年年底，平台开放淘宝服务758个，每天调用量19亿次，这一年SNS热潮消退，游戏逐渐淡出，卖家

市场依旧生意火爆，营销工具崭露头角，成为开发者新宠，淘宝客成为开放新宠（这一年返利网和团购一样火，只是前者收钱，后者烧钱）。

就在开放平台前景一片大好的时候，出现了一个让开放转变和收缩的导火索，一家做营销工具的公司“团购宝”每天凌晨都会通过接口同步客户设置的一些优惠商品信息到淘宝网，结果那天凌晨，微博上突然有很多人说某些店都是一块钱的便宜货，要知道这种事情在微博盛行的时代，传播速度之快，影响之大，当即很多卖家商品都被1块钱拍下。最后发现是线下的商品价格不知道怎么全被修改成1块钱，然后凌晨一同步，就导致出现了上面的一幕。从那时候开始，开放平台的KPI中增加了一个重中之重的功能：安全，包括后面的很多技术产品都围绕安全展开。此时第一个被波及提升能力的系统就是流式分析集群，20分钟一轮的数据分析要求压缩到3分钟，同时数据量已经从每天8亿条增长到了每天19亿条，花了两个月时间断断续续地优化集群结构设计和单机处理能力，对于其中经历的内容，有一天我翻Hadoop的优化过程时看到了相似的场景，具体就不在这里赘述，详细内容可参考Blog。简单地说，包括四个方面：充分利用多核能力用计算换内存；磁盘换内存，用并行设计来保证整体业务时间消耗不变甚至减少；Slave Shuffle来减少Mater的合并压力；数据压缩减少数据传输消耗和内存占用。

另一方面，由于2010年对于Jetty7的充分理解和封装，此时看到了又一个新技术的契机，2010年去美国参加Javaone，当时看到有老外用Jetty7的特性来实现Comet功能，Comet原先主要用于CS结构的应用搬到互联网上，因为不能用TCP的长连接，所以不得不用HTTP的长连接来替代原来的模式，同时国外开放平台也关注很多新型的API设计，其中就有Twitter的Streaming API，这种通过HTTP长连接方式推送消息到外部ISV（独立软件开发商）的模式引起了我们的注意。因此，我们决定将Jetty7上的封装近一步升级，支持Comet长连接方式，后端通过事件驱动的模式主动推送内部消息给外部，避免外部轮询业务接口。这个设计最重要的一点就是如何用最有效且最少的线程来守护多个长连接，支持到后端事件驱动的数据下行，如果给每一个长连接支持一个数据推送守护线程，即时性自然最高，但代价就是消耗众多空置连接的守护线程（详细内容见Blog）。这种模式刚出来的时候，从上到下都是质疑声，觉得太不符合常规做法，常规做法就是pull，认为开发人员无法接受，稳定性一定不靠谱。经过2011年的“双十一”，当天几个“尝鲜”的开发者用一台PC就支持几百万笔订单的高速处理，就让很多人明白了，技术要敢想，代码要敢写，细节要敢专，没什么不可能。也就从这以后，多样化服务TQL、Schedule API、ATS从开放平台的土壤上都长了出来，为更多的场景和更多的终端提供了各种解决方案和创新实现。

2012年：垂直化。这一年到现在，平台开放淘宝服务900多

个，每天调用量为25亿次，这一年淘宝客由于公司方向转变热潮消退，无线乘势而起，新业务（机彩票、酒店、理财等）、P4P、数据类服务都开始运营API，开放平台开发者的客户群体也从C店卖家增加到了B的品牌商和渠道商等。

这是一个业务多变的一年，这也是淘宝内部对开放平台认可的新阶段。第一个阶段是放任不管，任由开放平台部门开放和封装。第二阶段是由业务方负责支持开放业务，但开放后的结果概不了解，也无所谓了解。第三阶段就是业务主动要开放，开放后开始运营服务，培养ISV市场，带动业务的正向发展。

这一年由于业务量的增长以及分析需求到用户纬度，因此，在2011年年底启动了流式分析集群重构升级的项目，将新的分析集群项目命名为Beatles，希望它能够象甲壳虫一样，小虫吃树叶，再多都能吃下。2011年底到2012年初，用了近两个半月的时间做了一次完整的重构，将那么多年的补丁经验和老代码重新设计和实现，并且将Mater根据业务可垂直切分，最终解决Master归并压力的问题，当然期间的技术优化点也不少，因为我们的目标从3分钟压缩到了1分钟，而我们的数据量翻番，统计纬度细化到了用户纬度。（意味着结果也会很大，如果不靠文件做中转，如何来实现需要更多的分拆和协同设计）

这一年起了两个比较创新的项目：JS SDK和无线SDK（IOS，安卓），这两个SDK的出现在一定程度上由业务和安全两方面决

定。首先，2011年年底启动了社区电子商务化的项目，也就是现在所说的轻电商（XTao）项目，将更多的网站和淘宝衔接起来，此时网站间的融合就要求更轻便和简易，最成功的案例就是Facebook，于是2012年年初的时候，拿这FackBook的JS SDK一阵看，就开始动手写了，期间很高兴拉了UED入伙，这才使得这个JS SDK变得更加靠谱，更加专业。同时有了JS SDK，买家的服务安全性有所保证，因为原先的REST调用在授权以后是无法知道是用户发起的还是服务器发起的，而JS SDK从一定程度上还要校验Cookie的有效性，可以部分保证用户的在场和知情。而下半年的无线SDK，就是苦读一个月的各种文档，然后就开始动手玩儿了，由于对Java语言、动态语言、脚本语言都有比较多的使用，因此，Objective-C语言上手并不是那么困难，同时没有涉及过多的MVC的内容，做SDK基础层的东西还是比较得心应手的，就这样IOS的无线SDK版本就生成了，此时在开放平台的技术团队内部正在执行一个叫做Hack project的活动，其中一个自主项目就是安卓的SDK，因此，一个月后，安卓的SDK顺利诞生了。这两个无线SDK所担负的职责就是把控无线安全问题，不仅是淘宝，业界其实很多公司都还没理解无线开放的风险到底有多大，OAuth2基本就无法保证无线的用户安全，因此，如何在SDK和服务端融入更高级别的安全设计，成了无线SDK诞生的第一个重要需求。

另一方面，开放平台安全体系的构建成为2012年的重点，从两个角度对安全做了全方位的控制。

第一，用户。用户授权更细化了授权操作范围（细粒度到了数据范畴），授权时长。所有的信息可监控、归档、快速定位，我们内部叫做Top Ocean，简单说来就是对所有的访问日志做归档，归档的载体是块状文件，归档时对块状文件的所有记录按照需求建立索引，然后保留索引，上传本地文件到远端分布式文件系统备份。实时的监控服务调用和应用访问，授权异动。

第二，第三方应用。采用监控集群对所有ISV的服务器做安全扫描，对普通的Web安全漏洞做扫描，对应用的可用性和响应时间做监控。同时，正式启动“聚石塔”项目，提供弹性计算和存储能力及可靠的安全网络环境给ISV，帮助ISV提供自身应用的安全性。

至此为止，5年左右的技术历程已部分展示在了大家的面前，这些只是5年中比较有代表性的一部分，同时技术的发展也只是开放平台的一部分，前5年是技术变革带动开放平台发展，而接下去的5年将会是业务变革和理解带动开放平台的阶段，对业务的理解直接决定了开放平台的价值所在。前面轻描淡写地介绍了5年来不同开放业务的兴衰，其实这背后却有更多耐人寻味的故事，而5年后的今天，淘宝的格局为：集市（C2C）、天猫（B2C）、一淘（电商搜索返利入口）、无线、新业务、O2O（本地生活）、团购平台（聚划算），这些平台的价值是什么？如何找到自身定位？如何借助外力发展？如何面对流量入口的兴起、传统互联网企业的电商化、电商平台的竞争？这些才是开放平台2012年及下一个5年的精彩所在。

第四部分

我在淘宝这八年

2011年12月8日那天，有同事恭喜我，我才知道自己在淘宝已经七周年了。很多人问“七年痒不痒？”老实说，也曾经痒过，但往往都是一痒而过，然后又投入到水深火热的工作中去。回家之后就想，我在这七年到底收获了什么，且不论成败与否，这七年的经历，是我人生的宝贵财富。

第一年（2004年—2005年）

@王牌七公曾经说过，要是写一本淘宝的历史书，一定有很多人感兴趣，其实我也很想写写这样一本书。2004年12月8日入职的时候，我和@衲子如幻一起进来，迎接我的是骆冰和黄裳@岳旭强 ，骆冰是百阿的班主任，黄裳是我的师父。当时还没有百淘，先参加了百阿，百阿给我发了一本书叫《完美商店》，介绍的是eBay的故事，看的时候我就想什么时候我也写一本关于淘宝的故事来。

我进淘宝非常偶然，当时只是看到这个网站做得不错，自己也不想继续做对日外包的工作了，就过来面试了一把。刚进淘宝的时候，我被震撼了，它跟传统的企业非常不一样，到处都是生机勃勃的样子，还有前台的@香香的好朋友笑得很亲切，之前见到的公司前台的态度都是冷冰冰的。@武当三丰给我两张笔试题做，后来居然通过了。财神面试我的时候，问我为什么到这里来，我说很欣赏这家企业发展这么快，这样的企业一定有很多高

手，跟高手在一起成长一定很快。我每说一句，他点头“嗯哼”一下，以至于后来我跟我老婆形容公司的CEO时，她只记得那个喜欢“嗯哼”的人。

来淘宝做的事情是做Java开发，但是之前我只做过3个月的Java项目，连Eclipse都不熟（我来之前恶补了一下快捷键的操作）。一开始做事是在@岳旭强手把手的指导下做的，当时非常依赖旁边的几个老员工——多隆、正风、进宝、我行、不同、范禹、天川。做的事情也没有一个成熟的流程，常常是大家在论坛看到有人需要什么功能，我们问问PD是不是需要做，然后就写代码，提交给测试人员（给自在、郭芙、宝驹），最后就让青青打包，让剑英发布。我还很清楚地记得做的第一个需求是，有卖家说不想把货卖给信誉为一颗心以下的买家；有卖家不想卖给某个省市的买家，我就给卖家一个工具，让其限制某些买家不能买。于是我就在发布商品的流程和生成订单的流程中，找到合适的地方，加了几个合适的参数，写了几段合适的代码，就发布上去了，但是这个功能一直没敢启用。直到3年之后，2008年要拆分Denali的时候，这段代码都在，但都没有发挥作用。

做完第一个需求后，自己写的代码在系统上运行了，一下子有了信心。当时HR成立了百淘的项目，我先去百淘二期干了几天（本来要我参加一期的，但当时忙，延迟到了二期，现在百淘已经过一百期了），回来就投入到一个更牛的项目——支付宝。

支付宝一开始生存在淘宝系统上，但到2004年年底的时候，老马的慧眼看到了支付宝的未来。当时请了Sun公司的人把淘宝的第一个PHP版本变成了Java版，之后，就让他们做独立的支付宝系统。我和天川从淘宝派出来做支付宝和淘宝相关的业务，当时除了Sun公司的人和淘宝的人，还来了一位标志性的人物@fenng ，他是这个项目的DBA，记得他刚来杭州时，家里要装宽带，运营商服务不周到，被他骂了好久。后来（2010年）他在微博上大战中国电信已经相当有经验了。项目组中还有另外几个让我非常佩服的人——鲁肃、苗人凤，二人后来成了支付宝的首席系统架构师和业务架构师。就这么跟着这样一帮牛人干了3个月，支付宝在五一节的时候上线了。还记得几个DBA在做数据迁移时候的囧样，数据结构已经面目全非了，@正牌七公、 @fenng、多隆、鲁肃搞了三天三夜。我们熬夜的时候还有杭州的卖家跑来看我们，有一位是璧君 ，后来直接加入了淘宝。

当时那热火朝天的岁月令我至今难忘，我三天三夜没回家的时候，我老婆还打电话过来问“你到底还爱不爱我了？”我说：“怎么会不爱呢？”

进淘宝的第一年，我的级别是P1，现在已经没有P1了，后来调整过，我当时进来应该是现在的P3，记得那年年底的时候，三丰给了我4分的评价（超出期望），然后升级为P4，那是我成长最快的一段时期。

第二年（2005年—2006年）

做完支付宝，公司举办了一场硕大的庆祝仪式，带我们一帮人去了千岛湖，玩得很爽，但我有点不安，因为我与他们在一起才3个月，实际上只做了3个功能，一个是创建支付宝交易的接口、一个是接收支付宝订单状态的接口、一个是绑定和解绑支付宝账号的功能，而其他牛人设计出了一个巨大的系统，我处于深深的拜服中。半年之后，淘宝网组织的郊游又去了千岛湖，后来，另外两个项目的庆祝也去了千岛湖，以至于我都认识那边的

道路了。

从千岛湖回来，真正苦逼的日子开始了，我很长一段时间都在做善后的工作，就是支付宝系统中的一些问题。由于支付宝和淘宝是两个独立的系统，系统之间的通信是一个大问题，而银行与支付宝也需要通信，于是问题就经常出现：用户在银行付款后，未必能通知到支付宝，支付宝收到通知后，未必能通知到淘宝，于是用户的钱没了，淘宝的系统上却显示未付款，很让人崩溃。我和鲁肃尝试了很多种办法，一开始用MQ，但并发量上来之后老丢消息，消息的时间顺序也会错，后来他做了一个消息中间件系统，这个就是淘宝的Notify的雏形，现在Notify一天能发送几亿条消息通知，能保证通知到，也能保证不重复通知，还能保证消息有次序，相当不容易。

三丰看我在支付宝方面做了很多事情，而且跟鲁肃他们合作得这么好，还以为我水平大有长进，于是在维护着支付宝接口的同时，我开始了做PM（项目经理）的生涯。据说，我是淘宝的第一个PM，这让我很爽，我在做PM的过程中与SQA一起整理出了《项目管理流程》《PM工作手册》《系统设计模板》等开创性的工作，有些东西沿用到了现在。但后来有人提出敏捷概念的时候，我又在反思，我是不是误导了淘宝的项目开发模式很多年？

我的PM生涯从2005年持续到2008年，这三年又大致分两个

阶段，一个是“新手上路”阶段，自己摸索着做了“商品详情拆分”、“收藏夹改造”、“支付宝认证”几个项目；另外一个是“死去活来”阶段，做了“我的淘宝AJAX版”、“招财进宝”、“淘宝旅行”。为什么是死去活来？因为后面这三个项目死了两个，活了一个。

“商品详情拆分”是在2005年开始做的，三丰说让我当项目经理，我看到“经理”两个字就吓得半死，但后来知道该项目组就我和拖雷两个人。要做的事情非常简单，淘宝商品信息表一开始就是一张表，商品的所有信息都在这张表中，包括商品的详情，用一个clob字段存储。大家应该知道商品的详情是多么恐怖的一个字段，据说，如果把淘宝商品详情页面打印出来，平均约5米长，虽然当时没这么长，但也很恐怖了，它与其他信息放在一起严重影响数据的读写性能。于是我新建了一张表来存储商品详情，普通的商品查询不会用到这张表，只在商品详情（detail）页面才会用到，做完之后，性能一下子好了很多，我又骄傲了很久。再后来我知道商品的详情已经不能存放数据库了，放到了文件系统上。

“收藏夹改造”是一个稍微大点的系统，最早，淘宝的收藏夹只能收藏商品，我和万剑、领军等人把收藏夹改造成能收藏店铺、收藏搜索和把收藏的内容分类处理，当时的UI设计可能是承志@sharkUI 做的，看他一个像素一个像素地扣，才发现他比我们

写代码更苦逼。这个项目算是一个比较完整的项目过程了，我除了写代码，也做一些工作计划，所以后来就开始写项目管理的文档了。

“支付宝认证”是淘宝的一个创新，淘宝在成立之初就要求卖家实名认证，最早的认证方式是让用户上传身份证照片，我们去连接公安系统的网站来核对信息，核对一个要交5元钱，成本相当高。后来浅雪@浅的雪过来做PD，提出了一个新的认证方式：我们认为银行一定有用户的身份信息，而支付宝又与银行有合作，那就可以通过银行的用户信息来验证身份。所以支付宝认证的原理就是：用户提交身份信息和银行账户，我们往这个账户里存钱，存进去之后，用户填写收到了多少钱（我们号称存过去的是1元钱以内的金额，实际上只有几分钱），如果用户填写的与我们向里面存的是一致的，那么这个人的身份就是对的。这不仅降低了认证的成本，也使认证的效率由原来的一周左右变成一天以内。由于我对支付宝比较熟，又做过PM，就理所当然地做起了这个项目的PM。据说，这个项目后来申请了专利，这的确是一个很大的创新。

这三个项目我做得很顺利，认为自己已经称得上是项目经理了，但巨大的失败在后面等着我，由于后面那些事情想起来太伤心了，先写到这里吧。

第三年（2006年—2007年）

2005年年底的时候，我结婚了，与老婆匆匆领了证就往公司跑，因为当时我负责了一个更大的项目，重构“我的淘宝”。由于太匆忙，结婚证掉在了路上，后来有人送到了民政局，结果是一天去民政局领了两次结婚证。

“我的淘宝”是给会员管理自己的商品、交易、收货地址、评价、投诉的地方，这个地方由于登录之后才能看到，所以风格与外观完全不一样，很长时间都没有优化过，样子丑，用户操作

也不方便，如果一个人有很多商品，上下架还需要一个一个地操作，非常麻烦。这时候一个重要人物登场了——承志@SharkUI ，他给我们演示了最牛的前端交互技术，就是Gmail上那种AJAX的交互方式，可以拖动，可以用鼠标右键，也可以选择组合键，操作完毕不刷新页面，管理商品犹如神助。除了承志，还有一个牛人加入了项目组——一灯@喻策 ，这是他作为PD的第一个项目。再拉上万剑和一伙工程师，我们就开始工作了，我给这个项目取名为alps，就是阿尔卑斯山，淘宝的前台项目叫Denali，后台叫Mckinley，都是名山，咱们这个要更有名。做项目的过程中，我把电脑桌面换成巍峨的阿尔卑斯山，加班的时候给兄弟们买阿尔卑斯糖，大家干得热火朝天。快完工的时候，老马不知道怎么回事，突然出现在我身后，看我操作了一遍新版“我的淘宝”之后，问我这是不是客户端软件，我说是网页，他抓狂了，说这跟客户端软件一样，链接下面的下画线都没有，上下架用文件夹表示，他都不知道怎么操作，卖家肯定也不会玩。被他这么一说，我们虽然不服，但也留了个心眼，于是做了一个beta版的发布，在老的版本之外让一部分用户先体验新的版本。

接下来，淘宝历史上第一个群体性事件爆发了，试用完新版本的“我的淘宝”之后，很多卖家愤怒了，说不会玩儿，一灯就和承志一起商量如何把页面改得像网页，改了半个月，愤怒依然没有平息，我很无奈地看着这两个人在那里坚持，然后跟老板们商量怎么办。后来，我们到论坛上让大家投票要不要使用新版

“我的淘宝”，投票结果是一半以上的人反对。于是这十来个人做了三个月的系统被杀掉了。我当时只感觉对不起这帮兄弟们，也对不起自己领的这三个月的薪水，走路都抬不起头来。但这还不是最痛苦的，最痛苦的是我们下线之后，另外一拨卖家不满了，说这么好的功能怎么没有了？

虽然“我的淘宝AJAX版”挂了，但老板们也没说我什么，我自己诚惶诚恐地总结出了项目过程中的几大罪过发给大家，警示后人，缓解一下内心的歉意。到2006年“五一”的时候，另一个划时代的项目启动了，就是“招财进宝”（我苦逼地连续失去了两个五一节）。财神说要用最好的项目阵容，我被选中了，这下让我觉得我能划分到最好的员工之类，原来正受伤的心又痊愈了。这是一个商品P4P的系统，就是按成交付费。我们认为已经有很多卖家有钱了，但淘宝上这么多商品，他们很难被找到，他们愿意花钱让商品排在前面。我们允许卖家购买广告位，把他的商品按一定算法给出排名（类似于百度的竞价排名，这里不仅能看出卖家交了多少钱，还可知道信用、成交量、被收藏数量等，这个算法很复杂）。这是一个多么牛气的盈利模式啊！在这个豪华的阵容里面，乔峰@王煜磊是业务方，浅雪是PD，开发有黄裳、进宝、晓锋、长空，测试有自在、非烟，UED是青桐和@sharkUI，架构师是行癫，DBA是叶开。

在我们开发的过程中，乔峰大侠踢球断了腿，于是他SOHO

办公，经常在网络的另一头给我们打气。我当时第一次听说SOHO这个词，只是大家SO的是home，他SO的是hospital。这个系统进行得很顺利，但发布的时候，更大的群体性事件出现了，买家们质疑：你们不是承诺3年不收费吗？收广告费不是收费吗？然后我们的竞争对手趁机在各种媒体上展开公关战，甚至在系统中开发出了“一键搬家”，搬过去还保留淘宝信用这样的“大规模杀伤性”功能。然后，这个项目又失败了。项目下线的那一天，乔峰先哭了，财神说男子汉大丈夫要拿得起，放得下，晚上去虚竹家请大家吃个饭，算是项目结束了。饭桌上，他才喝了两杯，就醉了，我看大家都吃不下去，也不敢多吃。结束的时候，我和小宝送他回家，小宝力气大，背着他，我在旁边撑着伞，路灯下雾蒙蒙的，小雨一直下。

有人说这个项目的失败让中国电子商务的成熟延迟了好多年，但这个项目背后的人真的损人也没有利己。

接连两个项目都挂了，我反倒不怎么悲伤了，心态反倒轻松了许多，明白了一个道理：很多东西，不是你努力就能成功的，也许应了那句话“谋事在人，成事在天”。

这期间也做了很多日常性的需求工作，印象最深刻的是胖胡斐有一次给我提了一个需求，年底要做一个抽奖的系统，要求在抽奖人数能预估的情况下，系统能够即时开奖，发奖数量要均匀

分布在一天的时间内，发出去的奖品不能超过预设的值，也不能有奖品没发出。真会难为人，我想了很久才设计出一个算法，用随机数来做抽奖的种子，数字在某一个区间的时候算中了某个等级的奖，每个小时发几个奖有限额，发完之后，在这个小时内的数字即便在中奖区间也不算中，如果前面一个小时很快就有人中奖，则减小后面一个小时的中奖区间。这个算法后来被应用到多次抽奖活动中。

心碎的第三年就这么过去了，那时候我的级别已经是P5了。

第四年（2007年—2008年）

“招财进宝”挂掉的时候，乔峰哭得最伤心，有人拿收费这个问题来大举攻击淘宝，他说：“有一天淘宝要收费的时候，请让我来宣布”。现在他在淘宝商城，商城收费不知道是不是由他宣布的，但因为收费的问题，商城又经受了一次攻击，不知道是不是同一个对手策划的（本人想象力丰富，对以上言论概不负责）。

在这个项目中有一个技术细节值得说说，“淘宝商品详情页面”每天的流量在10亿次以上，其中的内容都是放在缓存里的，做“招财进宝”的时候，我们要给卖家显示他的商品被浏览的次数，这个数字必须实时更新，而用缓存一般都是异步更新的。于是商品表中增加了这样一个字段，每增加一个PV，这个字段就要更新一次，发布上去一个小时后，数据库就挂掉了，撑不住这么高的更新。数据库撑不住怎么办？一般的缓存策略是不支持实时更新的，这时候多隆大神想了个办法，在Apache上面写了一个模块，这个数字根本不经过下层的Web容器（只经过Apache）就写入一个集中式的缓存区了，这个缓存区的数据再异步更新到数据库。好像什么问题到了多隆手里，总能迎刃而解。

那一年发生了很多事情，多到我都记不住了，我只模糊地记得项目结束之后，组织结构有过一些调整，也来了很多牛人，尤其是UED的人，例如，@XX的三通（他和我都是一灯的四个门徒

之一）、@赵小马、语凝、圆心等。那个时候UED开始分交互、设计、用户研究、前端等工种，以前都是一两个人负责所有的工作。下面那个项目中，我们有了深入的合作，UED的这种分工对于PM来说，让我觉得项目周期更长了。

悲催的一年过得很慢，但是年底我却升到了P6的级别。

2007年春天，我老婆回老家生孩子，我在公司开始生我的孩子，就是下一个项目“淘宝旅行”。我之所以觉得这个项目像是自己生的，是因为我从最早的商务调研就跟进了。那个时候，支付宝给很多机票代理商做接口，发现这个行业有很大的利润空间，而且电子化和标准化程度非常高，就想弄出个旅行服务的平台来，但是支付宝的研发人员都忙着没空，于是跟淘宝谈合作，我们做一个集团版本的旅行服务，可以接入淘宝、支付宝，也可以接入B2B的系统，后来我们做出第一个版本的“淘宝旅行”是用支付宝账号登录的，跟现在的一淘接入的用户系统相同，当时很多人感到很奇怪：怎么不用淘宝的用户信息？老板们看我闲着，合作方也是我比较熟悉的支付宝，于是我阔别两年之后，又一次跟支付宝合作了。

我跟着支付宝的BD孙权、夏波波一起去拜访一家家的航空公司，谈合作方式和分成模式，我第一次坐在谈判桌上与这么牛气的国企谈判，只在他们问我能不能实现的时候做一下技术分析。国企中有不少有眼光的人才，他们希望做点创新的事情，但整个

体制限制太多，谈了半年都没有结果。于是我们又转向与代理商谈判，这些商人嗅觉非常灵敏，他们意识到有淘宝这么多的用户能接入，一定是一个很大的市场。他们都很积极，于是“淘宝旅行”的模式很快就做成代理商作为商家的服务平台，代理商赚钱后与我们分成（分多少？不告诉你）。

艰苦的商务谈判完成之后，有更艰苦的项目开发跟进，老婆不在杭州，也不问“你还爱不爱我？”了，我专心投入到项目中，2007年年底，“淘宝旅行”上线了。上线的时候，我问老板们用什么域名，语嫣姐姐说用最土最好记的，于是jipiao.taobao.com就发布了。这个平台上产生的第一笔交易是一个代理商的老板给自己买的机票，按照分成规则，我们赚了几块钱。那天下午，财神和语嫣带了一波运营的同事，敲锣打鼓地给我们发金币巧克力，告诉大家这是淘宝第一笔收入。当时我被一拨人推着走，拍的合影也没人发我一份。这个项目的PD是小玉，运营支持是叶青和文纨，交互设计是语凝（从那个时候开始，我跟更多的女同事结下了深厚的情谊，所以我开玩笑说打算写一篇外传《我生命中最重要的十二个女人》，呵呵！），开发团队是震北、空望、清虚、腾翼（他们是《我生命中最重要的十二个男人》之一，呵呵）。

当时国内机票市场的老大是携程，老二是艺龙，我们的平台上线之后很长时间内，用户还是习惯打电话找他们订票，“淘

宝旅行”的发展速度其实是很慢的。另一方面，为了获取实时的航班信息，我们必须获得中航信（中国民航信息网络股份有限公司的简称）的支持，而中航信当时正急于推广他们的酒店系统。于是在接下来的半年时间内，我们用中航信的酒店系统做了“淘宝酒店”，跟机票一起纳入“淘宝旅行”，中航信也把IBE（Internet Booking Engine）接口给了我们。我和做运营工作的姑娘们整天盯着有谁在我们这里订了机票，有谁订了酒店。如果有同学、同事或亲朋好友要旅行，我们就给他们推销“淘宝旅行”。记得到2008年年底，机票销量才几百张。

坐在我们办公室旁边的是“淘宝彩票”的团队，哲别是PM，一灯是项目经理，经过了“我的淘宝AJAX版”之后，一灯技术水平精进，跟彩票公司谈合作的时候大谈AJAX和用户体验。这个项目真是天时地利，用户只需要两块钱就可以买一注，用支付宝下注非常方便，上线之后销量猛增，到年底的时候，有两注彩票都中了500万元，一时之间，风光无限。苦逼的机票团队在一边只有羡慕的份。

看着这个孩子在慢慢长大，在这个过程中，我们做了零零星星的一些日常需求，实际上工作量不大。当时另一拨人被关进了湖畔花园，做了“淘宝商城”，他们从湖畔花园回来之后，“淘宝旅行”这个垂直市场的项目被划分进了商城，然后我们的团队被并入商城的技术团队。但每次跟行癫开周会的时候，他们讲商

城的种种事情，我都插不进话。到了年底，商城蓬勃发展，但机票的业务没有太大起色，由于我的技术水平没有多大进步，结果年底被评了3.25分，P6级别不变。

这时候有人来挖我了。

第五年（2008年—2009年）

有读者担心，写到后面会不会变太监了。其实，越往后面越难写，一方面是那些人就在你旁边，你要顾及他们的感受；而那些事，也才刚刚过去或者正在进行中，身在其中，很难有个客观的描述。不过既然都写了这么多，那就继续写下去吧，后面的事情比较近，也不太有名，估计感兴趣的人不多了。

我曾经写过一篇博文，是对于“小黑屋”的描述，淘宝有个传统，很牛的项目都要在小黑屋中进行，当年我们做“我的淘宝”和“招财进宝”的时候，有人羡慕我被关在一个小黑屋里面。到了2008年的时候，我开始羡慕别的关在小黑屋中的人了。这样的人有两拨，一拨人做了传说中的“淘宝商城”，一拨人做了传说中的“淘宝系统3.0”。做淘宝商城的那拨人暂且不说，淘宝主站系统在那个时候经历了一次翻天覆地的变化。

因为发展实在太快，淘宝的系统一直处于变化之中，但大的版本变迁大概有两次，一次是把最初那个LAMP架构的系统迁移到Java平台上，一次是把集中的Java系统拆分成多层的分布式系统。从PHP到Java在2004年就迁移完毕了，我进淘宝的时候正赶上迁移完成后Sun的工程师开始做支付宝，所以我错过了。从集中的Java平台拆分成多层的分布式系统时，我在做“淘宝旅行”，又错过了。

我一直相信，像“淘宝旅行”这样的垂直市场才是最好的业务模式，不可能所有的商品用同一种展示方式，也不可能所有的交易用同一个流程。但在主站拆分之前，要做垂直市场只能重新做一个系统，拆分之后，只需要在业务逻辑层重组一下即可。所以这次网站的拆分不仅撑住了不断上涨的流量，也支持了后续几年新业务的发展。在拆分的过程中有很多技术的创新，我们从使用技术到了创造技术的阶段。记得有一次我们与腾讯的工程师交

流，大家提起那个时候各自做了一个文件存储系统，仿照GFS，淘宝做了一个TFS（Taobao File System），腾讯做了一个TFS（Tencent File System），在GFS的理论基础上，各自有不同的创新。除此之外，还有Key—value的缓存系统、消息中间件、Java中间件、海量数据的存储和计算等。一个系统发展到10亿次流量的级别，你不得不做技术创新。

这些我都错过了。

但“淘宝旅行”最初两年的发展非常缓慢，商城在做业务的创新，主站在做技术的创新，我们这个团队游离于商城和淘宝主站之外。这个时候，有人来挖我了。

我总是容易被女人说动，尤其是漂亮的女人，而郭芙就是这样的女人。她问我淘宝主站拆分之后，对测试有什么挑战？我说系统分层多了，出错的概率大了，但功能测试无法探测到下层。她问有没有办法深入到代码级别来测试？我说有，但很难做。她说难才找你嘛，有什么办法来做？我说做单元测试，但单元测试最好让工程师自己去做，我们做再往上一层接口的测试。然后，她说英雄所见略同，他们已经有几个人在做了，问我愿不愿意一起把这事做大。这时我发现她挖了一个坑给我跳。不过这是淘宝项目开发中一项很大的变化，如果做好的话，将对系统的稳定性有很大保障，而我也觉得每天在那里卖机票有点乏味了，那就弄点没人弄过的事情吧。

我做过开发，做过PM，兼职做过PD，还像SQA一样做过几个流程，在2008年年底，我又开始了测试工程师的生涯。进到测试团队之后，我发现以前对测试的认识都太肤浅了，尤其是淘宝的测试团队，其测试技能和测试方法是业内顶尖的，我要做的就是招募一个能写Java代码的有测试思路的团队，但后来发现没有这样的人，于是招募能写代码的，然后给他们培训测试方法，另外，也吸收能做测试的，给他们培训写代码的技能。在2009年上半年，我大半的精力都放在了招人和培养人上面，团队从4个人增加到19个。我记得跟铁花一起做接口测试工程师的培训，有个外号叫老鸨的，给我们取了个名字——“花柳组合”。

接口测试的思路很简单，就是用测试代码来验证系统代码的逻辑是否正确。但做起来很难，最大的困难就是被测代码太“拥抱变化”了，三天两头地变，测试代码经常会失效；另一个问题就是要验证一个业务逻辑，也许要用10倍的测试代码才能覆盖，所以，这事儿也是一个体力活。我们发现系统越往下层，变动越小，逻辑越简单，于是我们就从最底层的IC、TC、UIC开始测试。测试代码写完之后，放入持续集成的环境中，一旦被测代码提交SVN，测试代码就回归一遍，把错误信息发布出来。2009年是这个团队异常艰难的一年，我们把最底层的代码都做了接口测试，有些还有性能的测试。我记得做UIC接口测试的时候，模拟了10亿次以上的请求量，结果发现了JDK中的Bug，并提交给了Sun公司。做这些事情的过程中，我们也对常用的测试方法做了

一个抽象处理，弄了一个测试的框架，写了一本《接口测试白皮书》。

我跟郭芙说，既然我们做了这么多事情，有这样的实力，就可以高调一点，向外传播我们的影响力，于是我们提出了一个愿景：“做测试的行业标准”。她任命我们“花柳组合”一方面去维护博客（地址为qa.taobao.com），一方面对内开展大量的学习和培训，我们叫它“3T交流会”（Taobao Test Technology、Taobao Test Team或者Taobao Test Training都行）。

在测试团队做得很开心，也有机会跟更多可爱的女孩子共事。但从很多人的眼光来看，一个开发的人员去做测试好像有点奇怪，我也常常思索自己这么跳来跳去到底好不好。直到前面一段时间看到网上流行三种青年的说法，我也把工程师大致分了类：普通工程师跟着业务跑，来啥需求做啥事；文艺工程师专注于自己的领域，研究得非常精深；2B工程师跳来跳去，啥都干，啥都浅尝辄止；还有一种工程师叫牛B工程师，啥都能干，啥都精通（这种人在工程师心中叫“神”）。而我应该属于第三种，不过幸运的是，淘宝里的机会太多了，我做的都是自己喜欢的事情。

2008年年底，我的级别从P6提升到了M1，悲剧的是，过了一年之后，公司更加重视专业能力，即M1==P6。

也许是看我在测试团队把培训做得风生水起的原因，2009年

年底，又有人来找我了。

第六年（2009年—2010年）

作为一个2B工程师，我渐渐地开始不务正业，到2009年就很少写代码了，只做一些上通下达、资源协调、关系处理、甩手掌柜之类的事情，完成了一个P到M的转变（从另一种意义上说，我这个码农废了）。但做了M（管理）之后，才知道管理真的是一门艺术，尤其是管理一群女孩子的时候，简直是处于艺术殿堂的巅峰。那时候从一位大侠那里听到一个理论，作为M，就不可能做到让所有的人都喜欢你。这对于一贯喜欢做好人的我来说，很

有难度。但做了两年M之后，有人说我是个“老好人”，这真是一个很大的打击，说明我这个M没有做好。

据说是因为有人推荐，2009年年底，老板的老板的老板来找我，我以为出啥大事了，原来是他们发现团队大了，壁垒也大了，知识的传播和传承有很大障碍，需要一个专业的技术培训团队。而我，啥都干过，又喜欢张罗些培训和交流的事情，似乎挺合适的。我认为团队的成长是M的第二等大事（第一等是干好活），那跟带一个小团队来比，支持整个大部门的成长似乎更能发挥我的余热。但这个我一手建立起来的团队，大部分人都是我找了无数简历、打了无数电话才找到的，我投入了太多的感情，实在不舍。思前想后，心理斗争了很久，明白大家都能独当一面了，我的离开对他们没啥损失。于是挥一挥衣袖，我走了，不过还是没忍住眼泪。

这一年的测试生涯使我仿佛又处于最初的创业时期，理论、方法、技能、团队都是从无到有，跟测试团队中其他人员的配合也是从生疏到默契，所有的事情都是摸索着在做。这一年，我的技术和管理水平没多大提升，却犯了很多错误。团队成员之间出现问题的时候后知后觉，处理人员关系的问题简单粗暴，工作的安排像是甩手掌柜。但这一切过后我收获的是心态变了，慌乱之后开始变得从容。

接下来我们新的团队成立了，@展堂、@早安薇薇安、@奇

怪的伟大是最早入伙的。我们取了一个很霸气的名字“@淘宝技术大学”，还有英文名Taobao University of Technology，定位是做一个企业大学，像惠普商学院和摩托罗拉大学那样，但是要做什么其实也不知道。我们看每年都有几百名应届生入职，而很多技术都是外面学不到的，那就先给他们进行培训吧。

于是淘宝技术大学旗下第一个项目开始启动了，我们叫它“逐浪堂”，取长江后浪推前浪之意。我们收集研发部中所有的业务、框架、规范、流程、工作方法，教给这些新人。这些知识收集上来之后，我们发现需要有两个月的时间才能培训完，于是逐浪堂前面几期的同学接受了我们两个月的知识灌输。然后我们去访谈他们的意见，发现一个最大的问题，就是东西太多了，如同把天山童姥的功力传输给一个路人甲一样，他会崩溃的。当然，这些同学也有不少是像虚竹这样的，本身天资聪慧，可以接受这些东西，成长很快。

2010年，我们把主要精力都投入到了应届生的培训中，“逐浪堂”项目几经修改，我们对知识分门别类，对课程精益求精，最终成型：应届生接受两周的“逐浪堂”脱产培训，内容是通用技能；进阶版的技能放入在职培训，叫做“追风堂”；经验分享和大师交流叫做“百家讲坛”；还有一拨社会招聘的员工，他们参加3天的脱产培训，叫做“飞云堂”，主要传授淘宝特有的技术和框架。支撑这些项目开展的是一个讲师管理机制、一个课程管

理机制和一个在线报名的培训平台。

在开展工作的过程中，我们也有不少创新。兄弟公司的培训管理员曾经说过一件事情，他们找培训公司为他们采购一个数据库的课程，培训公司的人说：“最好的DBA都在你们这里了，你让我到哪里去找？”，我们才发现其实很多业内顶尖的人才都在公司内部。那我们能不能像采购外面的课程一样从内部讲师这里采购课程呢？在得到老板们的支持之后，我们开始花重金在内部悬赏这样的课程，把某项技术讲透，需要8个小时以上的时间，需要有良好的课程设计和授课技能。悬赏发出之后，一下挖出了十来门这样的课程。说是重金，其实比他们去外面讲课的价格还是要低很多，而且这样一整天以上的课程，对讲师来说是工作之外一个很大的投入，给些激励也是应该的。于是他们讲过几堂课之后，拿着我们发的iPad，既感觉荣耀，又帮技术大学做了很好的广告。

在工作迅速开展的过程中，团队也在不断地壮大，这时候又来了@唐甜cr、@风云咧咧嘴，中间也有几个实习生走来换去，其中一位的真名被我们注册成班主任的小二账号，到现在都成了花名。另外，还有一个兄弟团队需要提一下，因为我们之间的关系比较亲密，我们做内部的培训，他们做外部的交流。迄今为止，他们已经成功举办了两届@velocityChinaWeb性能和运维大会，也举办了D2、iData、iTest、aDev等很多交流会。他们主外，我们主

内。这个团队的负责人是@灵小珊，一个简洁利索的姑娘。

2009年，我刚从测试部门到技术大学，没有晋升，我也觉得的确不够资格。在2010年年底的时候，我充满信心，那时候淘宝的晋升机制变了，需要对一个晋升委员会做汇报，很多人从述职面试回来都深受打击，我也是其中一个。老板们问了我一个问题“你觉得培训的本质是什么？”我说这是一个好问题，结果是我继续留在了M1的级别上。

第七年（2010年—2011年）

这一期先打广告，哈哈。

首先帮@胖胡斐推销他的新书《玩法变了》，这是一本讲

述淘宝店怎么运营的好书，用一个很俗的词来形容，就是“干货！”，书中提到抽奖的玩法，其中有一次活动就是我写的代码。在“魅力属性”这个篇章中也出现了我的名字，嘿嘿。

然后再打一个广告，就是老包宗曦翻译的《触动人心》，主要介绍苹果上的用户交互，也是精品。我在淘宝遇到的牛人很多，但心甘情愿拜倒在他的牛仔裤下的神人不多，而胖胡斐和老包属于神人这个范畴。我和胖胡斐一起学车，我们倒车都是回头看车尾，只有胖子扭不过头，他是看后视镜倒车的。另外，这个家伙过圆饼总是会压到边，被@华黎曾宪杰和教练嘲笑的都快懊恼了。宗曦之所以被我佩服，主要是这家伙玩啥都能玩出境界，他对摄影的研究非常深入，“宗曦观片会”带出了很多摄影爱好者，他不在淘宝了，这个组织还活跃至今。

最后就要为我们的团队打广告了，第一年，我们对淘宝技术大学是摸索着玩，主要是培训了几百名应届生。第二年，也就是2011年，是我们全面开花的一年。先说说出场的人物：@展堂、@早安薇薇安、@唐甜cr、@奇怪的伟大、@风云咧咧嘴、@DY-段燕、@高小糕、@Ciera米_米……还有几个没有微博的人，以姑娘居多，而且个个可以说是才貌双全、色艺双绝。

在2011年，队伍壮大了，能做的事情也多了，年初我们做了需求调研，针对核心系统和底层产品开发出了一系列课程，有

《云计算系列》《海量存储系列》《Oracle系列》《MySQL系列》《操作系统系列》《小众语言系列》《JVM系列》《Java内存调优》《搜索技术系列》《广告技术系列》等。也根据这两年公司技术战略的几个关键词：稳定、性能、成本、用户体验，开发出了《稳定性系列》《性能优化系列》《用户体验系列》课程。还根据日常的热门技术做出了《秒杀系统的设计和优化》《双十一背后的技术体系》一道又一道的技术大餐。

在这里也有必要提一下，这两年淘宝的技术团队飞速发展，我对几位老板们也有了敬佩之情。技术团队成立了“技术委员会”执掌技术的发展方向和人才的评估，成立了性能、稳定、成本、用户体验等各个小组，招募了专业能力非常强大的人才，分管重要的指标，使整个网站系统的发展越来越健康。这两年，淘宝的技术也逐渐走向开放，有了面向开发者的开放平台，也把自己的核心架构和中间件都开放出去了，甚至把我们做的低功耗服务器的硬件结构都变成了开源的。淘宝前些年从开源社区获得了不少技术，现在我们真心实意地开始回报整个互联网行业了。

同时，我们也向前延伸我们的触角，与公司的HR一起走进学校，在浙江大学成立了浙大淘宝俱乐部，在大连理工大学成立了大工淘宝俱乐部，并送去了几个优秀课程，也给同学们做了我们提出的实验性项目。我们也跟着集团，与ACM中国区组委会一起举办了夏季论坛和预选赛，让学生亲近了企业，也让我们的研发

部门亲近了优秀的学生。

2011年上半年，我还纠结在“培训的本质是什么？”这样的问题上。想的多了，做的少了，主要是姑娘们在干活，她们发挥了强大的能量，组织了四百多次培训，反馈和辅导的讲师也有三百多名，给我们研发部每个工程师提供了差不多14个小时的培训。我从《ASTD美国培训和发展论坛2009年度报告》中了解到，美国培训做得比较好的企业中，人均培训时长在19个小时左右，而天朝的企业中被统计到的有培训的平均时间不到5个小时。而从培训管理员人均产出量来看，我们超过了美国2009年的数据。

在我讲概念和数据的时候，姑娘们已经开始在研究细节了，研究桌椅怎么摆放比较合理、学员怎么邀约会有比较高的出席率、讲师在课堂上容易出现哪几类问题、PPT的字号多大在后排能看得清……她们把培训的工种逐步细化，每个环节都有操作指南，也有了很多知识的沉淀和经验的总结。姑娘们，你们很棒！

培训的本质到底是什么呢？经过老板提点，我们认为培训的本质是：“通过知识的流转，促进员工的成长，进而推动公司业绩的提升。”那又有老板会问：“你们怎么证明自己的工作提高了公司的业绩？”这是一个好问题，呵呵……有一天我听到农夫山泉的一个广告，说“我们不生产水，我们是大自然的搬运工”，我灵光一闪、再闪……“我们不生产知识，我们是知识的

搬运工”。我们搬得越多，别人接收得越多，我们的价值就越大。我不知道我这瓶水是拯救了饥渴的生命，还是呛到了谁的肺，但我知道它一定有价值。

随着理论的补充和姑娘们的努力（@奇怪的伟大别介意，我们把你当姑娘了），在年中特殊晋升的机会中，我升了一级，变成了M2。

老马说过“唯一的不变就是变化”，年中晋升之后，公司有了翻天覆地的变化。淘宝一拆四，变成了淘宝网、一淘网、淘宝商场和共享业务平台，我成了共享业务平台的人，然后共享业务平台又拆分，我成了集团的人。跟我们一起变的还有成立不久的“产品大学”和“运营大学”，三个大学顺理成章地统一了，变成了“淘宝学院”，以前的校长成了院长，我就荣升为技术大学的校长。人多了，可以集中资源办大事，但层级不需要这么多，而我做的这事说是P或者M都可以，而且一个被人说是“老好人”的M估计也没大前途。于是我转了，级别变成了P7，头衔是“培训专家”，属于专家和教授的级别了。

七年了，我的头发越来越少，付出了7年的青春，也得到了不少，除了金钱，最珍贵的就是认识了一帮牛人，结交了几个老友，有了一段足以吹牛逼的经历。

第八年（2011年—2012年）

2013年的第一场雪（2013年1月3日，杭州来得比以往更早一些，那天下了一天的雪，我就知道第二天没法开车上班了，决定去等公交车，结果等了半个小时挤不上去，我就打算徒步去上班。想想8公里路的路程，准备试试，但走了两公里之后，我就走不动了，于是遛进路边的星巴克，打开电脑，领导在网络那头说要写一下去年的总结，我就好好总结一下吧。

2012年可以用百转千回来形容，在阿里巴巴，你永远无法预料下个季度你在做什么。2011年淘宝网一拆为四，变成了淘宝、

天猫、一淘和聚划算，我成了阿里巴巴集团的人，从业务部门变成了HR。2012年年初，马总提出了“One Company”的概念，要把集团内的资源打通，很多老板来回转岗，很多业务也统一整合，于是我的老板换了，老板的老板也换成了HR的副总裁。但老板上任之后，刚刚请我们吃了一顿饭，又换了一个老板。新老板是一个有策略的人，他不像其他新老板一样先来三把火，而是三个月内不点任何火，只是不停地带我们吃饭，观察我们做事的状况。

在这样的状况下，其实我们自己也在不断地寻求变化，本来技术大学、运营大学、产品大学刚合并没多久，我们也经常相互切磋学习方式的创新。我们认为线下培训的方式只是我们的第一版产品，随着公司规模的扩大、办公地点的继续分散，这个成本会越来越高，最经典的是在每年七八月的时候，能够顶着烈日走到一千米外的另一幢楼里上课都需要很大的勇气。还有一个问题是随着业务的拆拆合合，每个业务对培训的需求是不一样的，统一集团层面的线下培训无法满足小团队自身业务的学习需求。

那么我们的2.0版本的产品应该是什么呢？我们认为应该是在线学习，在线学习是一个说起来简单、做起来很有难度的事情，在线学习大概有四种模式：视频、文档、直播、问答。视频要解决拍摄的清晰度和声音的纯净度的问题，在一般的教室里拍摄的视频，通常是投影幕布很亮，人物黑乎乎的，窗外的汽车喇

叭声、鸟叫声很嘈杂，要是讲师在白板上写几个字，镜头还需要移动拉伸，这对器材、环境和摄影师的要求都很高。我们和集团的“淘宝天下”团队合作，借用他们的器材，学习他们的拍摄技术，现在拍摄的视频效果已经超越家用DV了，呵呵。在线文档的难度不在于产生内容，而在于内容的吸引力，PPT很难承载有实质内容的东西，PDF太长的话又没有人看。视频直播我们和淘花网合作，用在线Flash播放的技术，这个相对而言比较成熟，但也受限于摄影机、教室环境等硬件因素。问答模式是我们模仿“知乎”的作品，即通过学员主动提问，邀请同事回答的方式来营造一个学习社区，这个难度主要是早期种子用户的培养和优质内容的推送。

虽然困难重重，但大家取得一致的认可之后，我们就开始推动这些事情了。培训管理员小姑娘开始学习操作专业的摄影机，拍摄完毕还要做视频编辑，投入了非常大的精力。2012年半年就上传了30多门课程的视频（基本上拍摄和制作一部视频至少需要两天时间）。在线文档模块中，我们有集团内最活跃的知识管理狂热分子，他一个人就上传了几千份文档。视频直播主要还处于试用的阶段，在重要的课程上，我们开通了所有的员工可以在线观看的直播通道，这些重要的课程包括正祥、毕玄、褚霸等人的最新研究。问答频道在“问题少女”小组的运营下，也出现了不少优质内容，还有“元芳你怎么看”等颇具社区氛围的内容。

这些新的学习方式的探索是一个长期的过程，搭建一个平台很容易，上传很多内容也不难，最重要的是坚持维护其中的内容和及时更新有效信息，我相信只要保持这种热情和精力的投入，在线学习的平台未来能够发挥出比线下培训更大的作用。说到这里，顺便提一个概念——知识管理，我不太认同这个概念，在传统行业，知识或许可以管理，但在互联网企业里，知识的更新迭代非常频繁，等你理出来刚要管的时候，它已经失效了。我认为互联网行业的知识不是要去管理的，而是要让隐性的知识显性化，在它的生命周期里迅速传播出去。我们不需要等它沉淀，只需要让足够多的知识流动起来，就能创造巨大的价值。

老板一边和我们探讨新的学习方式，一边开始响应“One Company”的号召，打通集团内部的培训业务，我们团队的三个大学出自淘宝网，另外，B2B和支付宝也包含“百年技术大学”，在技术大学下面也有应届生培训、在职培训等多个产品。我们这些大学之间有些“民间”的沟通，但从来没有打通过资源，也没有共用过一个平台，可以说是各自为战，现在都“One Company”了，自然也要打通一下。于是在老板来了三个月后，我们启动了一个“小三通”和一个“大三通”项目，“小三通”就是培训资源的打通，包括课程、讲师、解决方案的互通有无；“大三通”是培训平台的打通，培训的资源放进同一个平台里。我任“小三通”项目的PM、苏苏（就是《人人都是产品经理》的作者苏杰）任“大三通”项目的PM。

于是在2012年这个春夏之交的时节，我们“两岸三地”（钱塘江两岸，以及B2B、支付宝、淘宝三地）的培训人员开始了“三通”的工作，记不得开了多少轮的会议，在即将结束的时候，变化又来了——2012年6月20日，B2B退市，分拆成CBU和ICBU，原B2B的技术大学并入集团，于是从组织结构上直接就“通”过来了，人都在一起了。

“在一起”之后，就到了火热的夏天，关于HR的另一项重大项目开始启动了，就是招聘。我一边帮招聘的姑娘们审核宣讲稿的内容，一边接手了一个很洋气的项目“常春藤计划”。这个计划的主要思路就是，未来我们只招最优秀的学生，这些学生要培养成技术上的“高富帅”。那怎么培养这些学生呢？以前我们的成长方式都是野路子，放养在那里，长成什么样是什么样。我们就想这些野路子里长出来的人有没有什么共性呢？于是就寻访了一批进步比较快的员工，问他、问他老板、问他同事，找出了他们成长的规律，按照这些规律制定了一个新员工培养的方案，这就是“常春藤计划”，这个计划比较机密，在此就不公开具体内容了。

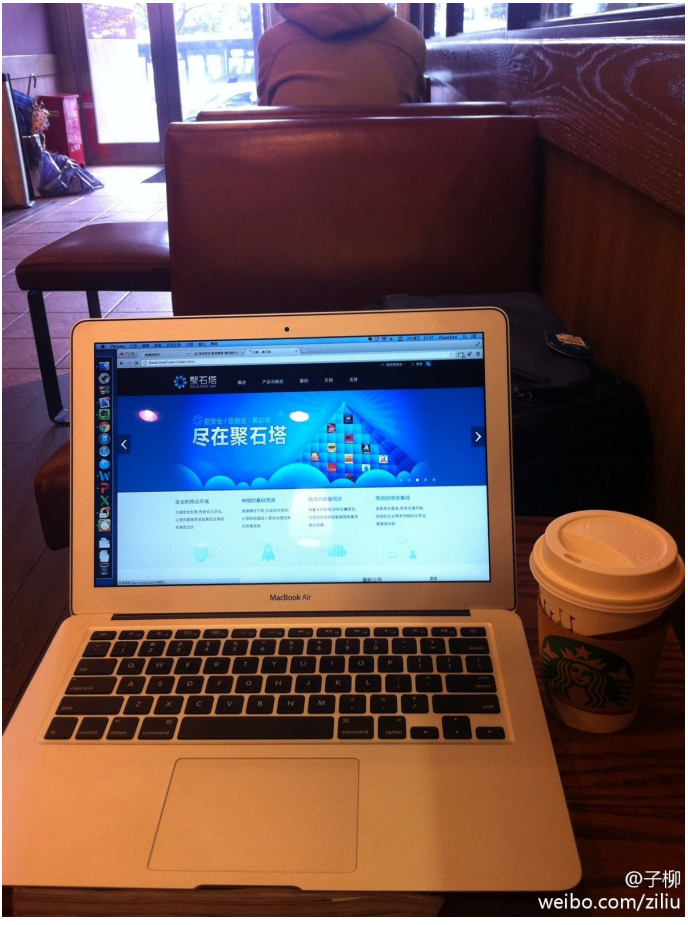
“常春藤计划”折腾出来之后，已经到了2012年秋天，负责集团内部创新项目“赛马”的姑娘也归到了我这边，她负责“赛马”的整体安排，我给她打打下手，偶尔替补一下评委，也为集团的技术创新和业务创新出了一把力。跟进了三个月之后有一些

感受，不管是外部创业还是内部创新，都面临着九死一生的压力，有想法没用，要有产品出来，甚至有产品还不够，要能够迅速推向市场。内部创新也面临着被抄袭的危险，同时还面临着“官方”产品的竞争。

写了这么多，其实发现这些事情我都是作为一个参与者或者支持者在做，这一年下来做的事情很杂，多数事情让别人作为主力就可以做好。我自己在支持这些事情的过程中倒是有些副产品取得了出乎意料的效果，为了帮问答产品做支持，有一段时间整天混“知乎”网，在上面回答了几个问题，居然进入了“「千人赞」俱乐部”，跻身知乎TOP 250（黄继新语）；业余时间整理了这几年淘宝的技术发展，被新浪博客推荐到IT头条，而且居然有很多出版社联系我出书了。

2012年的最后一个月，我一直在思索一个问题，在技术人员的成长方面，我还能贡献多少力量？答案让我有点失落，我离开一线岗位三年了，以前那些技术和知识也整理出来普及到大家了，对于学习模式的探索也有些眉目了，这方面有同事继续做下去就没有问题。如果我是一段木柴的话，在这方面已经燃烧得差不多了，那我就翻一面来烧吧。这三年错过了“云计算”、“大数据”等技术，也与淘宝上的商业渐行渐远，我想在这些方面补补课。而就在我思索这些问题的时候，天猫聚石塔的副总裁菲青来找我了，呵呵，聚石塔里面有我想要的一切。

（2013年1月4日）本来是我去聚石塔报到的第一天，却被大雪封在了星巴克，我就打开电脑在这里总结过去，展望未来吧。



第五部分

牛P列传

正明——集团核心系统高级研究员

正明是集团核心系统研发负责人，高级研究员。为2012年“大淘宝技术委员会”会长，毕业于国防科学技术大学计算机专业，工学博士，曾任TelTel首席科学家，国防科技大学计算机学院副教授，RedHat 内核开发者，LVS 开源软件创始人，ChinaCluster 的共同创办者。

子柳：首先恭喜你当选今年（2012年）“淘宝技术委员会”的会长。2012年将是技术委员会运营的第三年，在年会上也有一些委员疑惑的声音，例如，技术委员会是一个虚拟的组织，没有调动资源的权利，很难主导一些工作之外的技术创新，作为本届

会长，你对这些声音有什么看法？

正明：技术委员会在淘宝成立有两年左右的时间了，发挥了不少积极的作用，例如“Job Model”的编写、“晋升考核”、“技术赛马”活动、“淘宝开源”等，技术委员会确实做了一些事情。可能我们宣传得比较少，呵呵。有人提到去年（2011年）有个想法，但是跟业务线的关系不大，他去找那些主管，却要不到资源。其实像去年无锋做的“技术赛马”活动中的“taobao labs”项目，他也不是一个人做的，他找了好几个人，是跨团队的。那时技术委员会也出面，获得了菲青的支持，把外边的人员放进来，这样“taobao labs”得到了很好的开展。其实技术委员会是除了业务线之外的一个额外的渠道，在这里你能讲你的想法，拿到资源。可能这方面我们宣传不够，很多技术小二不太知道。2012年我们要在这方面多宣传一下，包括提升技术委员会在整个技术小二中间的影响力，例如，它有哪些作用，有哪些功能等。

子柳：技术团队在去年被分离成了几个部分，但是技术应该还是一体的，在大淘宝的大氛围下，“大淘宝技术委员会”在其中担任的是怎样一个角色？起什么作用？这些会给整个淘宝带来什么影响？

正明：公司组织架构有不同的划分，这是业务的需要。“技术委员会”是一个纽带，可以把各个组织架构下的技术部门串在一起。“技术委员会”是技术人员的一个组织，它不承担一个具

体的业务线，它的互动和学习的功能会更好一些。这样能够使整个大淘宝的技术人员形成合力，共同推动淘宝技术的发展。

子柳：作为大淘宝技术委员会的新任会长，委员会在未来一年的技术规划有哪些可以和小二分享呢?

正明：这个规划不是自上而下的，而是技术委员会的各个分会对自己的发展做规划，现在有7个分会，未来可能要增加一个“无线分会”，主要由这些分会来做规划，自下而上的规划会更有生命力。作为技术委员会的“常委”，我们主要的职责是申请到更多的资源和大家一起设立好的激励机制，让分会有更多创新的想法，更自主、自发地开展工作。技术总会上我们也会策划对内对外的交流平台，例如，技术委员会的开放日，我们还会搭建技术委员会的网站，开展内部交流活动（例如“空享会”），每个分会有自己的发展基金，“技术委员会”也设立一些奖项用来激励“技术赛马”、“技术创新”等。对外我们继续支持“技术嘉年华”、“技术沙龙”、“Velocity”等。

子柳：你个人的技术经历非常丰富，在读研究生的时候就开发出了LVS系统，然后在国防科技大学教书，接着去创业，最后又加入淘宝网，能分享一下你这一路是怎么走过来的吗?

正明：我是在念博士的时候开始做的LVS，2000年开始在国防科技大学教学，2006年正式离开大学了，先是创业了一段时间，

两年前加入淘宝。

我在1998年做的LVS项目，到1999年的时候，基于这个项目已经有很多应用了。那时侯我有个同学在上海，复旦大学有个MBA的创业比赛，他问我有什么想法，我就把LVS这个项目给他描绘了一下，结果他拿着这个想法去参赛了，结果好像没有获奖，哈哈。后来我们觉得既然写了这个商业计划书，看着还有点机会，就找一些朋友看能不能拿到投资。在2000年年初的时候，找到一笔种子基金，那时侯我还在学校里，我们成立了一个公司。当时正是互联网泡沫时期，我们也没什么经验，几百万元的钱我们一年就花完了。这就是我们第一次创业的过程，其实创业要考虑的事情很多，包括整个公司的运作、现金流、市场推广等。

（子柳：当时规模有多大？）规模最大的时候有40多人，在北京和上海两地。

（子柳：那是一个什么样的项目？）我们网站的名字叫chinacluster.com，即中国集群网络有限公司，围绕LVS做负载均衡的设备和技术服务。

这样差不多一年的时间，公司关闭了，我还在学校教书，做科研。2001年，我结婚了，2002年，我太太博士毕业了，想去北京，她是校子弟，从小在国防科大的院子里长大，从幼儿园一直

到博士没出去过远门，等毕业了就想一定要离开这个地方。正好我有个同学在中科院软件所，我们学校派我跟他们合作一个项目，然后就去北京了。当时我们看整个计算机产业的发展，一开始是大型机+哑终端，到现在是个人计算机的天下，这可以说智能从中心向边缘迁移，导致新的产业格局。然后我们看通信行业，当时整个通信体系把持在运营商手里，终端只负责接收和发送讯息。我们预见通信产业的智能也要往边缘移，网络的层面只要是个“笨网络”就可以，负责报文转发就行了（但目前来看，这个还是牢牢控制在运营商的手里）。当时AT&T的一个研究人员研究了十几年的智能网，但后来他把自己的理论颠覆了，认为智能不应该在中心，研究智能网都是错误的，智能应该在终端，我看了之后非常认同。他这个理念跟Bell实验室的研究是背道而驰的，后来他被解雇了。但我认为这是个潮流，一定会发生的，我们顺势而为，不求做出多伟大的东西，搭上这趟车就能成功，于是我们就围绕着这个主题开始研究。我们看到SIP协议实际上是解决了两个Peer（对等端）之间通信的问题，借助一个Session能够找到对方。这个协议跟我们的理念是契合的，我们就研究这个SIP。但是现实中有很多家庭网关，它是私有IP地址，要做NET地址翻译才能找到公网。SIP协议是七层协议，不会考虑到私有IP，这样两个peer都在私网中，怎么找到对方？NAT穿越就是一个问题。当时美国有一家公司Jasomi能做这个事情，一台Jasomi设备要卖好几万美元，很贵。我当时看了他们的规格，感觉也不是很难，正好北京非典爆发

了，我一个多月待在家里没啥事干，就把这个功能写出来了，没人打扰时的工作效率真高。然后我们在硅谷通过思科找到做SIP Proxy的一个小公司Cathay Networks，他们也快没钱了，我们就合伙干。我们有了SIP相关的技术，但不知道拿它们做什么应用。

我们做的一个应用叫Meet2talk，即聚在一起聊天，相当于一个语音聊天室或电话会议。我们这个是面向终端客户的，不是面向企业的，但是发现这个应用不是一般的人能聊得起来的，因为我们不想往娱乐的方向发展，因此用户很少。然后我们就想面向熟人做一个语音聊天工具，在2004年做了一个TelTel，推出来之后发现市场上有一个Skype，它是2003年9月做出来的，此时我们在技术上也碰到一些问题，面向大规模用户的时候顶不住了，后来美国这边的几位员工要离开，那边就做不下去了。当时我在国内给他们NAT穿越的支持，同时也在做别的项目。2003年年底，我们在国内做的一个项目叫比酷网（bitcool.tv），该网允许用户上传一些MP3、视频、Flash等，直接提供播放和下载，当时没看到Flash技术的巨大价值，这些多媒体文件都是用源格式播放的，流量很大，带宽的成本很高，也不知道流量的价值。由于付不起流量的钱，我们推出了P2P客户端软件aBitCool，后来收到好莱坞的律师函，又有了版权方面的问题。（问：差点做出一个youtube？），就差那么一点，若当初坚持下来就好了，呵呵。

美国的人要走，我们就去接美国语音通信的研发，解决了

TelTel大规模用户的支持问题，但发现做这个很难赚钱，PC上大家习惯了免费，电话那一段掌握在运营商手里（包括Skype现在也没有赚钱），我们根本没有议价能力。D-Link投资了我们一些，他们有很多网络设备，但是不知道用户是怎么用的，用我们的SIP技术可以得到这些信息。我们就跟D-Link合作，还做了SIP电话、SIP摄像头、SIP电子相框等，不过都是些小众的产品，也没卖出多少。再后来我们看中了互联网电视，做了一个9X9.TV（云端电视），即允许用户自己组织电视节目，自己做电视台，也可以订阅别人的节目，用我们的设备帮你下载内容，推送到用户的电视机上播放。由于广电总局一纸禁令不能将互联网的内容带到家里的电视上，最终我们这个项目在国内就很难开展了。

子柳：你到淘宝两年多来，一开始主要做LVS和HAproxy的大规模普及，之后做了很多CDN系统的改良工作，今年又推出GreenCompute项目。能介绍一下这几年工作的整体规划思路吗？

正明：这几年工作的整体规划思路就是为淘宝网打造一个高性能、高可扩展、高可用、低成本的基础平台，基本上是在这四个维度上不断深入优化。最早，淘宝的CDN用的是商用的调度负载均衡器Citrix NetScaler，这是当时业界最好的负载均衡器。但是淘宝因为规模越来越大，有很多针对小图片方面的访问需求，用Citrix就遇到很多问题，小图片造成的请求数特别多，万兆网卡的流量只能到3GB/s，一旦流量超过3GB/s，若到了4GB/s，系

统就会崩溃。在CDN系统中，图片处理的挑战最大，相比视频那种连续的数据，图片这种很小的、比较离散的数据对硬盘的访问要求很高。淘宝在2008年年底到2009年年初左右，曾经一度全用SSD，这样一个CDN的节点造价就会比较高，要花掉两百万元左右的钱，还要再加上商用的负载均衡器。而且刚才也说了，一个NetScaler流量只能提供到3GB/s，两台加起来也就只有6GB/s，而且两个NetScaler还不敢用心跳线，因为它虽然支持这个功能，但是万一有一台坏掉，6GB/s的流量完全转移到另一台，其结果肯定会令人崩溃。所以这就是商用系统的问题：在特别的负载情况下，它是不适用的。

于是我们开始逐步改造。用LVS+HAproxy，在硬件高配的情况下，一个节点跑到100GB/s都没问题。成本方面，后端肯定不能无限制地花钱，我们就开始用混合存储，SATA、SAS和SSD都有。经过优化之后，效果做到跟全SSD差不多，存储空间更大，命中率更高，像我们现在有些高的，命中率可以做到98%。那么现在我们一个最低标准的CDN节点，流量在10GB/s，成本已经优化到50万元；如果是低功耗，还可以进一步优化到37万元。这个项目我们以后还会持续优化，因为优化无止尽嘛，我们追求的目标是更好的用户体验，更短的响应时间，同时还要花更少的钱。

响应时间是我们最关注的指标，因为它直接影响到用户的体

验。其实淘宝目前在图片的优化方面做得算还不错，例如，今年“双十二”的时候，我们最高跑到了856GB/s的图片流量，这在世界上可能也是一个纪录了。你现在打开一个淘宝的网页，图片加载的速度还算蛮快的，目前图片请求的平均响应时间已经在10ms左右，但我们还会进一步挖掘，看能否达到9ms或8ms以下，让它加载得更快。虽然说，越往下面挖掘，难度会越大，但是这也是值得去做的。

另一方面，网络传输这方面往往也是瓶颈。一个图片请求发过来，硬盘的处理时间算为10ms，但是在网络传输方面，CDN部署的好的情况可能会占用20ms，如果网络有问题，就可能到70ms、80ms。所以，网络协议的优化方面，可以挖掘的空间可能会更大。

2009年年底，我们立项要做低功耗的项目，选择了当时我们掌控比较好的CDN系统。因为CDN对数据的安全性要求不是太高，毕竟上面都是缓存，数据丢了也无妨，全局系统可以随时把坏掉的节点切掉，对用户的影响低，所以这个低功耗服务器就专门针对CDN应用进行定制。这中间经历了很多事情，整个过程并不是很顺利，最终成功弄出来，也要感谢很多参与的厂商，像威盛、Intel，还有超微。总之，我们最终还是把这个基于Atom的服务器用起来了，而且性能优化效果与其他的Intel处理器相比，目前是最好的。其实低功耗服务器在整个绿色计算中只是一个小环

节，服务器还包含高性能服务器，服务器之外还有网络设备、机架、直流供电的电源，到整个数据中心的设计，都应该是绿色计算的范畴。而且我们现在定制的只是一款Atom D525的机器，实际上，低功耗也可以有很多款机器。所以，现在我们做的只是其中很小一部分，把它分享出来，也是希望大家都能参与进来，尤其是也在关注绿色数据中心的互联网企业。如果大家都能参与进来，对业界会是一个很好的促进。我们现在跟业界同仁已经在联系这方面的事情，目前正在进展中。

子柳：是什么机缘来到淘宝的？

正明：我在前些年其实做得很累，不断地试验很多新想法，商业的成果很难如预期。有时候有了不错的技术，却不一定能找得到用户，这很难创造什么社会价值。而淘宝的社会价值是显而易见的，系统的规模和挑战摆在那里，技术问题很快就能给用户带来价值。我在2007年到杭州参加过一次网侠大会，后来一直有人跟我联系，我手上那些项目结束之后，在家闲赋了一段时间就过来了。

子柳：你预测一下未来的淘宝网系统会有哪些挑战？

正明：未来规模的挑战会更大，数据的存储系统、中间件、CDN的存储系统，在规模上来之后，这些架构都要重新思考。像CDN现在已经有1000GB/s了，未来达到3000GB/s或更大的时候，

在不影响用户体验的前提下，如何降低带宽的成本，是不是要建不同规模的CDN节点，之间是不是要有点层次关系等。在后端的数据层，如我们的Hadoop集群真的到了10000台机器时该怎么处理？很多东西都需要重新思考。

子柳：你一直活跃于开源社区，你觉得我们从开源社区获得了什么帮助，我们又为开源社区贡献了多少？

正明：我们从开源社区得到的帮助太大了。例如，我们去年要提高数据库的性能，影响数据库性能的一个重要因素是存储的I/O速度。2010年夏天，褚霸和我们一起看最快的非易失性存储产品，就是掉电也不丢数据的。我们联系了国外做PCI-E的Flash卡存储的两家厂商，测试了6个月，效果很好。伏威的团队先在IC上把IBM小型机去掉，换上高端的PC服务器。然后我们帮着一起把存储换成PC服务器加PCI-E存储，也对数据库做了五六层的优化，从innoDB的存储引擎到I/O的调度器，再到Flash Cache和对文件系统进行调优。这样我们换掉了Oracle，去掉了EMC的存储，IC上去后效果很好。到TC的时候，一步就把IOE（IBM小型机、Oracle、EMC存储）替换掉了。淘宝在2010年的时候，每秒钟仅有2000笔交易，2011年原定目标是5000笔交易，换成PC服务器加PCI-E的卡之后，一台机器就可以做10000笔交易。过去的解决方案要花费2000万元，现在一台花费12万元就可以了。为了保险起见，我们放了一个16X2的集群，也只花费400万元就够了。在TDDL层拆分之后，单从数据库来说，一秒钟可以做160000笔交

易，这可以撑起淘宝未来很多年的发展。去掉IOE的成功，这对淘宝来说是一个标志性的事件。我们最核心的系统已经摆脱了商用软件，完全基于开源软件和自主开发的软件系统，所以说，开源软件对我们帮助非常大。

我们现在也把我们自己的一些软件开源出来，回馈社区，包括TFS、Tair、WebX、TEngine等。我们的淘蝌蚪平台上已经有了一百多个开源项目，这些项目不仅是淘宝工程师的，也有很多是外面的工程师贡献的。我相信国内的开源环境会越来越好。

子柳：你对刚入行的技术人员有什么建议？

正明：找到自己感兴趣的，花时间投进去，通过实践后的知识积累比只看书本有用得多。我看过一本操作系统方面的英文书，其中引用了一段中国人的格言：“I hear and I forget. I see and I remember. I do and I understand”，这句话给我留下非常深刻的印象。是荀子说的“不闻不若闻之，闻之不若见之，见之不若知之，知之不若行之。”

子柳：你在工作上一直很忙，我也看到你经常在微博上秀生活照，能描述一下你业余时间是什么样子的吗？

正明：主要是带孩子玩，我比较喜欢孩子，当然家里的事不是我做主，我主要负责陪玩，呵呵。

子柳：感谢正明抽出宝贵时间接受我们的采访，从正明这一路走过来看，总的感觉就是“技术驱动人生”，正明已经把技术融进了骨子里。

正祥——淘宝高级研究员，OceanBase项目负责人

他曾拥有众多响亮的名头，他没有下属，但却做着令人高山仰止的事情，他笑称自己“没品位”。下面让我们一起去慢慢品味这位技术大侠——正祥。

阳振坤博士（淘宝花名正祥），中国计算机学会YOCSEF荣誉委员。1984年进入北京大学，大学只用了三年，硕士只用了一年多，24岁成为王选的博士生。1997年破格晋升为教授，1999年成为北京大学首批“长江学者奖励计划”特聘教授之一，先后获得北京市科学技术进步奖一等奖、国家科学技术进步奖一等奖（排名第四）、第六届中国青年科技奖、北京市五四青年奖等。曾先后担任方正研究院副院长、北大计算机研究所副所长、联想研究院首席研究员、微软亚洲研究院主任研究员、百度高级科学家等。现担任淘宝高级研究员。

这样的名头足以叱咤江湖，也许你脑海中会浮现出某位成功可以复制的人或者某位人生导师的形象。但见到他的那一刻，你脑海中的形象就会被颠覆。运动鞋、宽松的休闲裤、略有走形的衬衣，斜挂着工牌。瘦高，眼神清静，说起话来总带着谦卑的笑容。他就是正祥——阳振坤。很多人对他的着装风格印象深刻，仰慕他的潇洒不羁，他却说，那是因为他“没品位”，让大家别学习这点。

这位“没品位”的大侠却在做着令人高山仰止的事情，他成功地主持研制了方正第六代和第七代RIP（光栅图像处理器），为我国新闻出版事业做出不可替代的贡献。中科院院士、北大方正科技研究院院长王选教授这样评价他：“阳振坤研制了我国第一个页面语言解释器，在新一代RIP的总体设计、软件结构、关键算法等方面做出了关键性贡献”。而今，他在淘宝主持研发海

量数据库OceanBase（oceanbase.taobao.org）。淘宝的数据已经远远超过了单个关系数据库所能支撑的最大规模，而且仍然在快速增长之中。对淘宝来说，研发海量数据库已经不仅仅是出于成本的考虑，如果不采取这些技术，可能很快会因为机房数量、成本等无法支撑下去。

面对这些世界级的难题，正祥研究得怎么样了？这样一位牛人，他身上有什么传奇的经历？他对年轻人的成长有什么建议？他身上又有哪些八卦消息呢？

带着这些问题，我对他进行了一次采访，采访的开头有一个很有意思的细节，正祥带着一个空的矿泉水瓶，自己去饮水机那里打了一瓶水，放在脚边。

子柳：恭喜你，你主导的OceanBase刚刚获得本届中日韩开源软件竞赛优胜奖，这个奖项的意义是什么？

正祥：这是三个国家信息产业部联合组织的开源软件的一个促进会议（第十届东北亚开源软件推进论坛），奖项是“第六届中日韩开源软件技术优胜奖”，我们国家有三个软件获奖，都是企业做的，淘宝OceanBase排在第一。

子柳：OceanBase主要是为了解决什么问题？

正祥：关系型数据库理论的成熟在20世纪六七十年代，技术

和产品的成熟是在20世纪七八十年代，从成熟到现在，它的技术和理论没有太大的突破。但实际上，需求是在一直变化的，今天的数据规模对关系数据库提出了严峻的挑战。最近，很多企业开始做NoSQL的系统，其实这个NoSQL可以说“No SQL”或者“Not Only SQL”，我更倾向于叫后者，咱不能否定SQL。我们要解决的问题是数据规模的问题，数据库本质上是一个单机系统，即便是做了分库分表，这些也没有改变的是单机系统的本质。单机系统导致它的数据规模和处理能力都会有一定的限制，你要提升这些，只能用更好的服务器——小型机、中型机、大型机，它的成本上百倍地增长，但性能却没有跟成本一样提升，性价比很低。而淘宝的数据规模已经到了这个关口。

打破这个瓶颈，分布式的系统是一个选择，这方面的大规模应用最早是由Google开始做的，但我们的需求跟他们有一个不同的地方：搜索引擎可能索引不到一些页面，也可能索引到一些页面的速度较慢，但淘宝是从事电子商务的，用户的商品、交易都不能遗漏和出错，页面的展示要尽可能地快，我们需要更高的数据一致性，也需要存取速度更快的数据库。

因此，我们要解决的问题就是在淘宝的数据规模不断增长的情况下，提供高容量、低成本、高一致性、高可靠的数据存储和访问服务。

子柳：网友袁鹏-wugu123问，传统的关系型数据库的数据规

模受到制约的主要原因是什么?

正祥：关系数据库的数据规模受限的根本原因是目前的关系数据库尽管有各种方式的扩展，但本质上是单机系统。

子柳：OceanBase是什么时候开发完毕的，现在的应用情况怎么样?

正祥：开发完毕还说不上，因为这个系统还需要持续开发。我们是从2010年5月开始开发的，2011年2月发布了第一个版本。现在应用到了5至6个系统上，收藏夹是最早的应用，现在有63亿条记录，3TB的数据，访问量也比半年前增加了一倍以上。以前用Oracle的系统是可以换成OceanBase的。

子柳：竞争对手对OceanBase有以下看法，对此你怎么看?

- 有一个中心点，在数据量很大的情况下，单点有风险；
- HBase开源，大公司主导，很多坑都走过了。

正祥：OceanBase最好的地方就是具备事务，数据一致性很好。HBase在数据容量上会有优势，几千万亿字节都有可能，但它没有解决事务的问题。从数据规模来看，虽然有个单点(有热备的)，但OceanBase数据规模仍然可以达到关系数据库几十倍甚至几百倍的规模。

子柳：OceanBase推广的成本高不高?

正祥：OceanBase的推广应用得到了各个方面的配合，DBA团队已经跟OceanBase融合在一起了，OPS也是不遗余力，非常关键的是，OceanBase得到了应用和业务团队非常大的支持。数据库的迁移是有工作量的，而且还是有风险的。在应用部门的配合下，OceanBase已经做到了从原来的关系数据库平台无缝迁移到新的平台上，并且不停止服务，在对OceanBase进行升级时也不停止服务。到现在已经跑了半年，系统也比较稳定，且没有停止过服务。

子柳：OceanBase应用之后，节约了多少成本?

正祥：还真没细算过，首先Oracle的License就是很贵的，这个是我们能省下来的，服务器的数量也能减少。例如，收藏夹，原来每个机房有16台机器，OceanBase刚上线的时候是一个机房14台机器，后来数据量和访问量都已经翻倍了，还是14台机器。现在我们正在换成6台SSD盘的机器，预计能提供更大的访问量。

另外，成本最高的其实还不止是这些设备，而是网络带宽和机房机架等的成本。若减少机器的数量，会降低这方面的成本。淘宝的业务在高速增长，数据量和访问量在加速增长，但电力和机房资源不可能同步高速增长，从现在起，我们就必须在提升服务能力和性能的同时，降低服务器的使用量，否则不仅是成本令我们无法承受，机房和机架也无法找到。此外，从节省能源的角度来说，我们也必须要低碳环保。

子柳：在你的人生经历中，不断被破格提拔，你认为原因是什么？

正祥：我的运气比较好，机会也比较好。我自己的要求比较低，除了对住的地方要求高一些（我睡觉很轻，呵呵），吃穿之类的都没啥追求。我属于那种党叫干啥就干啥的人（虽然不是党员），而且我的运气非常好，遇到了一些很好的老师（比如王选老师等），站在巨人的肩上做事做事，把手上事情做好就成，就这样。

子柳：从你的经历来看，你对现在的技术人员的成长有什么建议？

正祥：很多人会说年轻人比较浮躁，其实我的身边有很多非常优秀的年轻人，他们聪明、刻苦、有闯劲、愿意接受新事物。年轻的同事想赚钱，想提升自己的职称，这些都是十分正常的。在这点上，我特别喜欢马总的理念——做公司要赚钱，但阿里从不把赚钱作为第一目标，我们服务好了客户，客户赚了钱，我们一定会得到自己应得的一份。在个人成长问题上也是类似的道理，这就是，一个人如果把做事、做成事作为主要目标，该他得到的东西，一定会顺理成章的、水到渠成地得到，但是，如果把上升作为主要目标，做同样的事，结果就会完全不一样。一句话，你的心态会最终决定你的成就。

子柳：在北大期间师从王选院士，他对你有什么影响？

正祥：我能做他的学生真的是运气很好，王老师给我的影响非常大，他心胸开阔，是我接触到的最伟大的人。我跟了他13年，从他身上学到了不少东西（可惜我悟性和能力不足，有太多的东西没有学到）。当时我们研究所只有二三十个人，他对我们的指导非常多。他和他的夫人陈堃銶教授前期做了大量的研究，才有后来我们团队做出来的那些成果。而那种朴素的只想把一件事情做好的感觉，也深受他的影响。

子柳：除了王选之外，在技术领域，对你影响最大的人是谁?

正祥：没有其他人了，王老师的高度太高了，没有人能像他一样，给我这么大的影响。

子柳：你经历了联想、微软、百度、淘宝这几大企业，能否描述一下你在各个企业中的主要收获?

正祥：跟着王选老师在北大方正集团做了十几年，当时的方正是技术好但管理不足的企业，而联想是一个管理很好的企业，我很想看看联想是怎么管理的，所以离开方正和北大后就去了联想做无线投影仪设备，前后用了差不多3年时间，完成后，在探索新的项目和方向时，就定位为存储，无意中了解到了Google的GFS，但云计算和联想主业（PC等）的差距较大，做这个事情的可能性较小。于是我去了微软亚洲研究院，做了一段时间后，发现要应用到微软那些比较核心的产品上的障碍很多，包括地域

的、语言上、文化的，做这个最好还是去硅谷或西雅图。一年多之后去了百度，在百度待了两年多，做的偏BigTable（当时HBase还不能用），百度是一个星期重建一次索引数据，当时就想希望改进这一块，在百度那边的项目做完了，但最后没有应用，并且项目也被关闭了。

子柳：以上几个企业的技术风格有什么不同？

正祥：联想要求研发人员要了解市场，跟着一个产品从市场需求到开发，再到小批量生产、真正量产等整个环节。微软的研发模式比较重，流程和审核机制非常严格，每一行代码都要审核很多遍，做事很稳，但很慢，我想这也是它在互联网市场很难施展的一个原因。我觉得百度其实不如淘宝重视技术，KPI导向的文化很重，各部门之间的协作和配合比较难（这一点淘宝要好不少），不同部门、不同项目的开发人员做了不少有差别但其实比较类似的东西，看起来个体效率高，但整体效率未必高，这可能是百度加班很严重的原因之一。

子柳：你很早就是一个研究室的主任，能够带队开发。但你在淘宝是没有下属的，为什么？

正祥：我在微软和百度也没有下属，我最大的长处是做技术，管理方面比我擅长的人有很多，那就让他们管吧，呵呵。管理对我来说实在是一项艺术，我还是更偏技术。

子柳：你在没有直接下属的情况下，靠什么方法带领开发团队？

正祥：其实事情是大家一起做的，这不是我个人的产品，是整个团队的，这个团队非常优秀，多隆在这个团队做了不少关键的工作，东邪、正明和楚材等领导以及整个团队都非常支持我。

子柳：你还在写代码吗？你怎么控制系统的实现符合你的预期？

正祥：我非常喜欢写代码，前段时间还在写一个数据压缩的算法，但效果不理想，没有放上去。现在写代码的时间很少了，但他们写的代码我都要看，这两个月在外面招聘新人，有些新的代码还没来得及看。

子柳：你还想在哪些领域有突破？你的研究方法是什么样的？

正祥：我还没有那么长远的眼光，但从Google来看，他们做GFS、MapReduce、BigTable的人都还一直在做，我觉得淘宝在数据存储和处理领域应该还有很长的路要走。例如，我们的事务目前是单机做的，但后面也可以做到一个集群，另外，我们也考虑把SQL语言的支持逐渐加进去。我们现在只是在传统数据库上迈了一个小台阶，后面还有很大的空间。

子柳：你怎么平衡工作和生活？

正祥：我对生活上的需求比较低，没什么特别的爱好，这个还好，就是在家的时间比较少一点。

子柳：能否八卦一下你的家庭状况？

正祥：三口之家，日常生活中都是太太做主。

子柳：你在着装方面很朴素，这种风格很多人都津津乐道，能否八卦一下原因？

正祥：其实我的出身比较贫寒，小时候真的是饿过肚子的。我小时候的玩伴，现在还有人在为生计发愁。所以我现在非常知足，我认为社会给我的已经高于我给社会的。我的品位上不去，这不是刻意为之的，大家不用学习这一点。

编后语：这位大侠系出名门（北大王选的门徒），也出身寒微（贫寒人家），他在生活上非常“没品位”，却在技术上追求最顶尖的水准。这些矛盾的属性在他身上却是协调的统一。这种自身的谦卑造就了他在事业上坚定的专注，对正祥的这些品味，也许值得从事技术的朋友好好品味一下。

毕玄——集团核心系统资深技术专家

毕玄从2006年加入淘宝开始，在商业产品中使用OSGi，先后编写了《OSGi实践》《OSG进阶》两篇Opendoc，以及《OSGi原理与最佳实践》一书，为OSGi在中国的推广起到了绵薄之力。目前负责淘宝网的高性能服务框架，此服务框架每天的服务请求量已超过100亿次，2010年6月出版了《分布式Java应用：基础与实践》一书。

个人感兴趣的技术方向为：高可用、高并发、高性能、可伸缩、低成本的互联网技术，以及动态化、模块化的大型分布式Java应用的技术。

子柳（主持人）：你做的HBase集群的应用现在是什么规模？

毕玄：现在大概有15个HBase独立集群，上面大概有20个项目在运行。现在还没有很大数量级的，都是几十亿条数据，还没有百亿或千亿条的。后面我们很快会上一个项目，那个项目应该是千亿条数据以上的。

子柳：淘宝的HBase有什么特色？

毕玄：最重要的是稳定性方面的改进，其他的可能与外界都差不多。在部署上，我们的HBase会加上一套监控系统和一套与运维相关的东西（若没有这个是很危险的）。还有一个，我们的HBase比官方的成熟一点可能是我们的应用比较多，不是一种用法，而是有很多种用法，我们面临的问题比别人会稍微多一点。

（主持人：这样会不会把HBase搞复杂呢？）不会的，我们跟Facebook一样强调只把HBase当作一个存储。

子柳：像你这个层级的技术专家，除了做好自己的东西外，还需要推广自己的技术和理念，在推广的过程中有没有障碍？

毕玄：其实从现在来看，推广的几个东西都还好。障碍方面，通常是看你要推广的这个东西跟其他的冲突有多大。可能当时正好有别人也要做类似的东西，这个时候也许能会有一些障碍。对我来讲，我不管别人有没有用我的东西，只要我的想法落地就行了，这个想法到底被怎么实现，谁去实现，其实没有那么重要。

子柳：做这种底层的通用的产品，需要跟很多人合作，你最喜欢和什么类型的人合作？

毕玄：只要能干活的人就好了，大家都认同这个想法会给公司或业务带来什么好处，然后就去做，不要想太多。

（主持人：这样说来，是不是有些不太干活的人呢？）是有一些人会比较看重自己的职责，如果不是自己的职责，可能就不管。

子柳：你在微博上写道“在公司四年以来，最让自己自豪、最有成就感和最骄傲的是三件事（一个产品，两个思想的推行和落地成产品），现在正在做第四件，计划第五件”——能不能解释下，这五件都是什么事？

毕玄：第五件不能说，其他四件可以说。第一个产品是淘宝的自动发布系统，其实这个系统的技术含量很低，现在淘宝所有的Java系统都是通过这个发布的，全部是自动发的。对我来说，虽

然这个产品没有什么技术含量，但它给淘宝带来了改变。这件事也是我自己发起的，所以很有成就感。

关于另外两个思想，第一个是容量规划，这个事情以前淘宝是没有做过的，虽然最后这个事情也不是我做的，但后来在CSP中也成了一个产品，我觉得容量规划对一个公司来讲，是一个非常重要的阶段。还有一个思想是优雅降级，优雅降级这个事情说了很久，一开始是很难被人接受的，到现在几乎所有的系统都采用了，这个目前最明显地体现在淘宝保护“大促”的一种情况。

现在我们正在做的第四件事情，就是把淘宝的整个运维体系做一定的修改，现在淘宝的扩容很不方便，会受限于网络因素等。以后我们会希望能有一套系统，使整个淘宝的扩容非常简单，你只要想扩容，就可以扩容，然后我们会尽可能地提高机器的利用率，这个是现在正在做的第四件事。

（主持人：我本以为你是要说一下Java中间件的方面）这可能是我在公司影响到人最多的一个产品，但我认为是因为刚好我在这个位置上，如果换做别人，结果应该是差不多的。

子柳：你自己经历了几次技术转型，为什么转型？转型最大的困难在什么地方？

毕玄：我觉得我的转型是跟淘宝绑在一起的。我刚到淘宝面临的是淘宝访问量带来的压力，我们需要解决访问量的问题，于

是我们就做了很多Java的中间件，其实当时我也不是很熟，呵呵。

当解决了一个访问量的问题之后，数据量就开始大增，于是就面临一个数据量的问题，我们就开始想怎么支撑这么大数据量的存储、分析、挖掘等工作，所以后来我就转型到做数据方面的工作。

转型最困难的地方在于，很多人以为我的层级很高，对我抱有很大的期望。但是过去的所有这些积累，对我的帮助并不是很大。当我进入一个全新的领域后，很多东西都要从头开始。但是由于大家认为我的层级很高，所以就期望我在很短的时间内能做出很好的效果，这会给自己带来很大压力。如果我转型做的这个产品刚好是这个时代所需要的，有业务场景，这样会好很多。

子柳：我了解到你是学生物学的，现在却是互联网技术专家，请描述一下这一路是怎么走过来的？

毕玄：我本科是学生物的，毕业之后是做政府企业系统软件的，这个跟互联网行业完全属于不同的领域。我在企业领域做过多年，后来有一些机会接触到互联网行业，比如腾讯的技术。这些互联网的技术对我们企业做开发的人的震憾是非常大的，而且成就感也不一样。以前我也会觉得互联网应用没有什么技术含量，就觉得那只是一个网站而已，搞技术的人都能随手做出来一个。但我了解到腾讯的技术以后，就觉得加入一家互联网企业也

挺好的，想去看一下一家互联网网站是怎么做出来的。后来我就加入一家叫做“五百万”的做彩票的网站。加入之后，我才发现一个好的网站真的是很难很难做，其中的技术含量非常高，有很多东西需要学习。从“五百万”网站出来以后，我就加入了淘宝。

子柳：你每天有多少时间用于编码？多少时间学习？

毕玄：现在如果没有什么特殊情况的话，我会花很多时间看技术方面的文章或者图书，写代码的时间可能每天只有两三个小时。每隔一段时间会去想一想将来干什么比较好，因为写代码时间会过得很快，要经常跳出来想一想。

（主持人：有哪些比较占用时间但价值不大的事情呢？）比方说，开各种各样的会啊……

子柳：你来淘宝技术大学授课的时候，给学员说过“但行好事，不问前程”，现在很多同学都因为晋升的问题很纠结，可否解释一下自己是怎么看待晋升的？

毕玄：大家都说了，晋升是一个“水到渠成”的事情。大家都会在意晋升，这个是正常的，除非你不在意级别，生活上也没有压力，不过这样的人是很少的，呵呵。如果真的没有升上去，这个也没有什么办法能够挽救了。我觉得重要的是在这个过程中你回顾了你一年做了什么事情，对公司有什么贡献，技术上有哪

些成长。

对于技术人员最常见的一种情况是晋升名单公布的时候，你去看谁升上去了，然后对比一下自己，觉得他水平不如自己，为什么是他得到了晋升，而不是你。我觉得最重要的是看那个人对公司做了多少贡献，你可以说你的技术确实很强，但事实是你对公司没有做出任何贡献。

子柳：你是技术晋升的评委，在评审的过程中，你比较看重什么样的特质？

毕玄：如果你是向技术方向发展的人员，我们要看技术方面的专业性；然后看你的技术对公司的业务发展有多少贡献。还有一点，我比较看重的是，也许你不在其位，但能够跳出自己的范围，想到公司未来到底会面临什么问题，用什么方法来解决。当然，仅想是不够的，如果你能够落实就最好了，我们不管你落实的技术含量有多高，关键是你解决了什么样的问题。如果你能够做到这些，你这个人对公司就非常重要。

子柳：你经常出去招聘学生，你欣赏什么样的学生？

毕玄：其实在校招聘的时候，我比较欣赏的学生往往是那些很“不务正业”类型的。我经常会问他们，你有没有利用业余时间出于自己的技术兴趣做的一些小东西。这样的学生我们通常会比较感兴趣，我认为这样的学生是真正喜欢技术。聪明程度一

般就可以的，他能够进入这些不错的学校，智商是不会有什么问题的。

子柳：而立之年，有房有车，马上又有小孩了，可以说“老婆孩子热炕头”都达到了，接下来你还想追求的是什么呢？你的人生理想又是什么？

毕玄：以前想的是用技术来改变世界，现在看来不得不接受这是很难做到的。所以，现在想的就是自己所做的技术能够给公司带来改变，使公司从一个阶段上升到另一个阶段，这就是我目前的理想。

放翁——淘宝开放平台项目负责人

放翁于2006年3月加入阿里巴巴，2007年初加入阿里软件创业团队，负责基础平台架构设计与实现，2007年年底与淘宝合作开始研发开放平台，负责开放平台体系架构设计与实现，2009年8月加入淘宝，负责淘宝开放平台架构。

主编寄语：

有这么一类工程师，他们在技术上有了深厚的积淀，他们初为人夫，也初为人父。他们既是公司里的顶梁柱，也是家里的顶梁柱。他们这一路是如何成长的？他们当前在承担什么样的责任？他们有哪些痛苦与快乐？他们如何展望未来？我们找到了这样一位典型的代表人物——放翁，且听他娓娓道来。

作为技术专家

子柳：作为一名技术专家，你的成长之路是什么样的？

放翁：我从毕业到现在，就在两家公司干过。第一家就是东方通信股份有限公司（简称东信），然后就是阿里巴巴。东信是国企，我在那里做了4年，从一个学生很快做到一个小部门的主管，这个成长的主要原因是我的直属领导不断离职，我就不断地补上去了。从技术上说，在这里有成长，但不是很大。4年以后，我感觉自己的发展遇到了瓶颈，虽然在这里生活比较安逸，但不

是这个年龄应该有的状态，当时也才27岁左右，就想换个环境。

然后我应聘了阿里巴巴的两个岗位，第一个是一个后台的部门，发现不是我想做的。后来有一个阿里软件的概念，提出了work at Alibaba的想法，当时自己对这个想法挺感兴趣，就加入了阿里巴巴。加入之后发现并没用想象中的那么好，因为一下子从一个工作灵活度比较大的人变成了一个从事最基础的编码工作的人员，有点失落。到了2006年年底，我被封闭到湖畔花园去做阿里软件的创业。当时收购了成都的一家公司，这个公司专门做建模类型的应用开发，这种方式的开发比做通常的Web应用开发能发挥的空间还要小。大部分情况下是用XML文件配置的方式来搭建一些管理软件的应用。底层已经封装起来了，这个工作比现在大家做的开发还要单调。这个时候出现一个问题是收购过来的公司只有3个人做核心平台的开发，然后是一大帮人做业务的开发。但这个框架本身还不成熟，有很多bug，业务开发人员就会提很多bug给核心团队的这3个人，他们当然是忙不过来的，于是大部分人就是等着他们去解决。我出于对这个框架的兴趣，就开始研究前端、后端和整个系统，然后我帮他们去修复一些bug。我没有提交主干的权限，每次修改完就告诉他们，他们审核没有问题之后，就提交上去。就这么两周以后，负责这个平台的老板就邀请我加入他们团队，于是我就调到了平台部。有这方面的能力，再加上机会，之后那3个人还是在支撑业务，而我负责整个平台的基础体系和架构的运营。从湖畔出来之后，整个阿里软件的4年左右

的时间，我会负责整个平台的发展、底层架构。所以，在这方面我会走得比较靠前。

后来跟淘宝开放平台合作，直接转入了开放平台这里，很多事情都是业界第一次做，做到现在快4年了，不断地把技术做深。

总结下来，我的成长就是从一个国企到一个公司，有一个落差。在创业团队找到机会，从创业团队中做到一个公司的架构师，然后再坚持把一个产品不断做精做深，最后才有一定的技术发展和影响力。

子柳：你在这个行业里差不多有10年了，在工作上有没有什么痛苦的地方？

放翁：会有些失落的地方，例如，从东信到阿里巴巴的时候，一下子很难适应，做阿里软件一开始也是做很多琐碎的事情。我也跟很多新人说过，其实来公司三个月到半年的时候，是最难熬的时间。接下来在阿里软件也痛苦过，虽然别人觉得这家伙很清闲，在做自己喜欢的事情，但是跟淘宝的现状一样，很多跟业务离得相对比较远的一些中间件团队做出来的产品，应用到业务系统上会有很大的阻力。在阻力面前，很多时候会屈服于业务，有些业务会要求开一些白名单、黑名单、特殊通道之类的东西，而作为基础服务，我又要保持它的完整性和统一性。这时就需要一些协调和沟通的技能。作为一个架构师，经常会有一种失

落感，有时候会发现我们做出来的东西有可能没有办法实施。我坚持做开放平台做了4年，后来我有一些感悟，我跟毕玄都发觉真的需要找到一个业务点，把技术做深，去解决一个个的问题，然后这个平台的效果才能体现出来。现在有了一个转变，从单纯地做中间件、平台架构，到成为业务团队的一个业务架构师。现在技术架构师和业务架构师两方面都在做。

开放平台是一个中间的东西，很多地方是需要依赖于后端，我需要后端支持的东西有时候他们并不关心，这时候就要坐下来与人谈。在公司里做事，很多人是自己做还好，但要跟别人一起做，跟各个部门协作的时候，就遇到麻烦了。对于技术人员来说，这也是需要成长的。

另外，跟其他公司做开放平台一样碰到的问题，就是每个部门都希望去做开放，这个时候开放平台响应够不够快，服务够不够好，会使得别人愿不愿意通过你的开放平台来做。我们发现百度、腾讯、盛大也都这样，开放平台希望自己是全公司统一的开放平台，但事实上是每个部门都在做开放平台。现在我们的思想也放开了，与其是通过行政的手段去堵，去要求别人，还不如自己把产品做好，这样这些部门自然而然会去想他要不要花这么大代价去做这个事情。我在想，做技术产品需要转变一个思路，靠行政和强制措施要求别人用你的东西，短期有效，但长期来看，还是要看技术过不过硬，有没有站在用户的角度考虑问题，产品

做得够不够好。

子柳：级别到达P级中最高的以后，你在技术上的目标没有人能帮你把握了，那你现在关注的重点是什么？你后面还要怎么成长？

放翁：其实从P7级开始，就没有人帮我做规划了。技术委员会在级别P7～P9的定义中，要求P7级的人员要对一个小的产品或团队有方向性的指导，P8级就要求在一个大部门或公司级的产品上有方向性的指导，P9级要求除了考虑自身的产品之外，还要站在公司的角度考虑自身的产品对公司的发展有什么帮助。对我来说，在开放平台不能只考虑开放平台本身发展得好不好，要看它对其他部门或整个公司的发展有什么帮助。我坚定地做开放平台，是相信那么多公司做开放，未来的合作多于竞争，例如，我们与新浪合作，他们能得到更多的微博用户，我们能得到更多的交易量。我想把开放平台做得更深入，能够跟各大互联网公司打通，借助外部的各种资源，给我们的买家和卖家创造更多的机会。

从技术上看，我都是贴近实际的问题来找突破点，解决了问题，技术就掌握了。说实在的，现在也会遇到一些瓶颈，到一定阶段，我的技术已经足够快了，但是业务上还没有跟上，这时很多技术人员会觉得困惑。我自己会去多想一些东西，包括我们现在去做TQL、长连接数据推送等，我们在技术上有很大创新，在业界也有很大的影响力。但站在客户的角度来看，我们需要用一

些方式去推广。明年（2013年）我会花一些精力去学校、ISV等地把我们好的东西推送出去。

子柳：现在开放平台做出了什么样的成绩？

放翁：从光棍节的大促销看，那些商城卖家和集市上的大卖家有95%都接入了开放平台，从实际效果来看，接入开放平台和不接入的有90%的差别。也就是说，他们处理订单的速度、发货的速度和退货的速度有非常大的差别。我们对大卖家的ERP系统已经了有很好的贡献，另外，我们对类目的导购、装修市场、手机淘宝，以及与新浪、美丽说等系统的结合做得越来越多了，整个体系在慢慢地打开。

子柳：腾讯、新浪都在做开放平台，我们和他们是一个什么样的关系？

放翁：对于互联网上普通的产品，大家做得越多，其竞争越厉害。而开放平台不是，大家做得越多，其体系越完善。比如，我们与新浪合作，大家各有所需，他们需要用户信息、活跃度，我们要的是交易。就像Facebook一样，它不是什么都要，他要的是用户，用户在这里玩任何东西产生的结果，像买东西、玩游戏，他们不关心，用户在他这里玩，这个平台就是成功的。

子柳：你已经升级为淘宝技术大学太师级讲师，你这种传道的动力来自哪里？

放翁：最大的动力是我希望开放平台是支持淘宝未来3～5年发展的一个平台，这样一个平台单靠一个部门做不了，我希望借助技术大学这样一个入口，把我这些思想传播给更多的人。当他做事的时候，会知道有开放平台这样一个方向。

子柳：你给工程师的分享反响非常好，你认为什么样的分享能打动工程师的内心?

放翁：有两点，技术人员首先会关心技术好不好，若技术不好，讲得再好都会感觉有些浮，所以每次我都会讲些技术方面的内容，不是具体的实现细节的技术，而是通用的一些思想和方法。另外一点就是我对开放平台的一种信仰和思想，我能通过开放平台为淘宝做什么。这样其他人会感受到听这个课是有帮助的。

子柳：你对新人的要求是什么样的?

放翁：第一个是做事要自己思考后再去问别人，而不是一遇到问题就找人求助。第二个是不断地打破自己的一些想法，你不要担心自己今天已经做了50%的工作，要是推倒重来，前面的事情都白干了。我现在带的两个新人成长很快，但是都有类似的经历，就是一个东西被我反复推翻重做，在这个过程中就是不断地成长，要思考我为什么让你推倒重做，若想不清楚，下次重做的概率会更大，这样慢慢地就会学会了思考。

子柳：你是CSDN的著名博客，你写些什么内容?

放翁：现在半年我会写两到三篇，写Blog的方式是当我做了一次深入的优化，或者有比较大的积累之后，我才会去写。现在跟以前不同，平常不会去写一些入门性的内容，因为这些大家都会了，网上也有很多。所以更新的速度比较慢，但我还是能保证质量。

子柳：你对工程师有什么忠告?

放翁：任何一个公司，不管用什么手段，都做不到绝对公平，最终只会有小部分人得到机会。这个时候去抱怨、愤怒都没有用的，只有自己不断地努力争取机会才行。

作为孩子父亲

子柳：你的孩子现在2岁半，有孩子之后，你的生活有了什么转变?

放翁：时间方面会有很大限制，我的生活方式也改变了。以前我是很有规律地早上做运动，晚上早睡。现在很少能在12点之前睡觉，运动也减少了。晚上下班要挤一部分时间给孩子，等他睡觉后，我才有时间做点自己的事情。

子柳：有了孩子之后，心态方面有什么变化?

放翁：有了孩子之后，我发现孩子很锻炼人的耐心，小孩儿会无理取闹，但我们不能跟孩子一般见识，要好好应付，这样我的心态会很好，这对工作也一样有益，但我在这方面转变得还不够。另一方面，有了家庭，我工作的意义就会变得不一样，我要为这个家庭负更大的责任，对自己的要求更高了。

子柳：你对孩子的未来有什么期望？

放翁：没有太多的期望，希望能身体健康，长大后能做自己喜欢做的事情。（你会推荐他干我们这一行吗？）这个看他自己的选择了，要是他喜欢，我会支持，若他喜欢做别的，我也一样支持他。

子柳：你把TOP也当作自己的孩子，这是一种什么样的心态？

放翁：开放平台正好与我的小孩出生的时间差不多，真的要把自己的产品当小孩一样养，我们不能生完了就扔掉，要为他的成长负责。不管小孩子有什么不好的地方，他的脾气会不好，身体会出问题，但是我们要有责任去帮助他成长，帮他去解决问题，不能用一种旁观者的心态去看。

作为美女的老公

子柳：作为一个技术男，你是怎么吸引到老婆的？

放翁：跟我老婆认识到结婚，也是一件蛮奇怪的事情。怎么奇怪呢？就是在东方通信的时候，有人给我介绍女朋友，是一个女孩子的同学，她帮我张罗，但我不喜欢她的同学，拒绝掉了。后来反倒越来越喜欢帮我张罗这位，然后就……哈哈。我老婆脾气很好，我很喜欢。要说他看中我的应该是我还比较实在一点，感觉比较可靠吧。

子柳：工作这么忙，交给家庭的时间会比较少，怎么平衡工作和生活？

放翁：平时家里有爸妈在帮着带孩子，所以减轻了很多压力，晚上不想让父母太辛苦，会花一个小时左右的时间带孩子，周末我会抽出一天的时间专门陪孩子玩。

子柳：老婆脾气这么好，会不会为琐事而争吵，怎么解决？

放翁：也会的，很多时候都是为了一些琐碎的事情而吵。我想这也是一种磨合，只要不是太过分，慢慢就会产生默契。我有一点不太好，生气的时候喜欢冷处理，就是不搭理人，时间久了也就好了。

子柳：老婆是网购达人，你起到了什么作用？

放翁：其实我来阿里巴巴之前，帮我老婆开过店（2006年开的），还帮她送过货，后来就不开了，所以这方面她比我还擅

长，不用我培养，本身就很熟悉。（反对你老婆买太多的东西吗？）其实她买自己的东西我没有什么意见，买家里用的有些东西我会反对，本身家也不是很大，买很多东西太占地方。

业余生活丰富多彩

子柳：你是一个K歌达人，也喜欢户外运动，这与很多宅男程序员不同。

放翁：以前读书的时候，没现在这么多娱乐项目，学校旁边有小歌厅，15元包一个小桌子，喝茶唱歌，台下有很多张小桌子，你唱一首，大家都鼓掌，这种感觉不错。很多程序员都比较宅，我觉得大家最好是去做户外运动，天气不好的时候去唱唱歌，也能够让你舒展身心，把压力也排解出来了。

子柳：作为土生土长的杭州人，给我们推荐点玩的地方吧。

放翁：在杭州，如果你不出去玩玩，就太可惜了。我推荐大家去走走云栖竹径、虎跑后山、江洋畈，这些地方的人比较少，很安静，能够让你放松身心。长一点的路线是从古荡上去到北高峰，然后到宋城。

子柳：现在你的家庭美满，事业顺利，你后面的人生追求是什么？

放翁：我曾经说过，如果有一天我离开淘宝，我会选择一个小公司或一个创业团队，把一个东西从小做到大。这样什么都经历过了，事业方面就满足了。然后我希望能到世界各地走走，这辈子也比较完整了。

吴翰清——阿里云集团信息安全中心高级安全专家

吴翰清，人称小黑，毕业于西安交通大学少年班，从2000年开始研究网络攻防技术，在大学期间创立了在中国安全圈内极具影响力的组织“幻影”。从2009年起，加入阿里巴巴云计算有限公司，负责云计算安全、反网络欺诈等工作，是阿里巴巴集团最具价值的安全专家之一。

子柳：见到安全专家，我首先想到的就是我的电脑是否安全，那么杀毒软件能保证客户端安全吗？

吴翰清：杀毒软件似乎是个人电脑的标配了，但这个只能做到“看起来安全”，要真正的安全，其实是非常难的。像流行的

木马之类的病毒，它第一个对付的就是杀毒软件，它在实施一次攻击之前，要看是不是通过了360诺顿、麦咖啡等各种杀毒软件，然后才会拿出来实施攻击。

子柳：这两年出了不少互联网安全方面的问题，企业在面对安全问题的时候都是非常害怕的，可能一下子就会被搞死，你认为企业需要花多少成本在安全方面？

吴翰清：其实安全一直紧随业务发展，在你开展业务的过程中，安全是一直要占一个比例的，但这个比例不能盲目扩张，要先做最急迫的事情。另外也要看行业，有些行业就比较容易遭受攻击，例如，那些卖假药的、网游私服之类的，这种是最容易被竞争对手攻击的。像地方论坛这种无利可图的就很少会被攻击。所以，要看你所处的行业和企业的规模来做安全方面的投入。安全的投入又分很多方面，例如，招人、买设备、买安全服务、做评估等。

子柳：安全方面的人才还很稀缺，有没有类似云存储一样的“云安全”服务？

吴翰清：我们是希望做到这样一件事情，现在在阿里云上面的多数是一些中小网站的站长，他们没有太多的钱投入到安全方面，例如，防DDoS设备，最便宜的都要几十万元，可能他们的网站一年的利润也就这么多，这个投入对他们来说就太不值。所以

云计算有个好处，就是把这么大的一个资源的投入切分成很多小份，然后按需分配。例如，如果你使用“云安全”的服务后，遭受多少次攻击，我们就帮你清洗掉，事后按次数收费，这个成本就降下来了，可能你只会“消费”掉几千元钱的攻击，我们把安全服务做成一个可以消费的东西。

子柳：作为一个黑客，在成为“白帽子”之前，是否都会有一段“黑帽子”的经历？（解释：简单地说，白帽子是做好事的，黑帽子是做坏事的）

吴翰清：我不知道现在的人是怎么样的，对我们那个时代的黑客来说，多多少少都会做一些攻击或入侵的事情，这个要看你的出发点是什么，如果你是为了找到网站的漏洞，或者是为了做技术尝试，进去之后不做破坏，这些我们认为是好的。（不过现在刑法修正案有了明确规定，这种行为也是违法的，所以不能提倡）另外一些人以利益为目的，这就涉及黑色产业链了，其实这一类人并不一定要掌握多深的黑客技术，有些是猜别人的密码，甚至先取得别人的信任，发给对方一个木马，这就不是黑客技术了，而是骗术。

我们在招人的时候就很小心，要看他以前做过什么，历史不清白的是不能要的。

子柳：前些年有中美黑客大战，有卖熊胆的网站被黑，有卖

良心药的网站也被黑，这些你都参与过吗？

吴翰清：我现在不太关注这些。往往是一些年轻人，他们比较愤青，看到这些网站就顺手试试，不能说他们的技术水平有多高，只是这些网站在安全方面做得太差了。

子柳：在阿里期间，你参与了哪些黑白帽子之间的较量？

吴翰清：很多人可能会认为黑客之间的较量很酷，大家你来我往，刀光剑影。其实这都是电影给大家的误导，现实中看来很简单，我对自家的网站做好各种防护，有人攻一下，进不来就走了，或者他们进来了，看到点什么东西，走了。有朋友向我们报告过漏洞，几个互联网公司的安全团队都是有交流的，有时候我们也会告诉腾讯、百度，他们的网站有哪些漏洞。

其实最头疼的是那些做黑色产业链的，主要是钓鱼和欺诈，这些人其实不能被称为真正的黑客，他们偶尔也会利用网站的一些漏洞，或者用木马，他们用的是半骗半攻击的形式。我在这方面一直投入了很大的精力在做，帮助淘宝和支付宝减少这些业务方面的安全问题。以前一个钓鱼网站出来，可能它能够存活半个月，没有人去管它。现在经过我们的努力，它出来几分钟就会被我们嗅探到，甚至他在旺旺上发一次网址，这个网站就失效了，他只能骗一次。

子柳：现在有没有比较令人头疼的水平比较高的黑帽子？

吴翰清：比较少，这个圈子里的人就几个去向，要么去大一点的互联网企业，要么去安全厂商，要么被国家队招去做国家安全方面的工作。

子柳：对于白帽子黑客来说，有哪些信条或原则呢？

吴翰清：首先要有职业道德，然后要假设一切都是邪恶的，我们自己的人也不能完全相信，要通过技术手段来保证，所有人的操作都有记录，做审计，然后收权限，我们自己也只有很少几个人有服务器的权限，保证我们自己也黑不掉公司。

子柳：在你的书《白帽子讲Web安全》中有很多攻防演示的案例，会不会担心被人学去做坏事？

吴翰清：任何事物都有两面性。首先，我写的内容都是互联网上公开的，我的每一个引用都能在互联网上找到。如果一个人真的想去做坏事，他在互联网上很方便就能找到。另外，就如同我们会说开源软件更安全，就是因为它的代码都公开了，我们都能看到其中的内容。对于黑客技术也是一样的，这个武器放在这里，攻击方案放在这里，你知道怎么去攻击，也就知道怎么去防御。

子柳：每一个新技术的出现总会有人找到漏洞进行攻击，攻与防似乎是自然界生态体系的基础，在互联网行业，这个生态平衡吗？

吴翰清：只要技术在不断地发展，就会有新的安全问题出来，这就像自然界一样，达到了一种生态平衡。（对于攻的人，他只需要找到一两个漏洞，对于防的人，要找出所有的漏洞，把它补上，这会不会很辛苦？）的确会很辛苦，而且有的时候必须跟随攻的人，他们在研究哪些漏洞，你也要去研究它。例如，黑帽子最近在研究手机，你就必须跟着去研究手机，等他研究云计算了，你也得赶紧去研究云计算，而且还要比他们研究得更透彻，赶在他们下手之前，把漏洞补上。

子柳：作为后台人员，系统出了问题才会想到你们，没有问题似乎就不知道你们的存在，工作的成就感来自哪里？如何评价自己的业绩？

吴翰清：我们确实为安全怎么量化这个问题头疼了很长时间，后来我们引入了一些体系化的东西，把安全融入到整个开发流程中。在软件研发流程中的每一个角色都要跟安全打交道，我们去培训他们要怎么做、不能怎么做，这样可以让他们感觉到我们的存在，呵呵。另外的一些量化手段包括今天我们发现了多少安全漏洞；到下一个月这些漏洞是变多了还是变少了；工程师写出来的代码是变好了还是变坏了。

我们的成就感主要体现在两方面，一是公司的产品是不是越来越好了？二是我们在工作的过程中攻克了很多技术上的难题，这个纯粹属于技术上的成就感。

子柳：你给出了一整套的互联网安全解决方案，但需要很多部门和角色参与，你怎么推广你们的安全技术?

吴翰清：也没有特别好的办法，我们专业的术语叫做SDL（安全开发流程），它依赖于整个公司的软件工程的成熟程度。这个公司的软件工程做得越成熟，越容易推动这些东西，在B2B有一些比较成功的实施经验，SQA会帮我们把每个环节都打通，我们只需要告诉SQA每个环节做什么。在其他公司有些开发过程还有点乱，就不太好推，我们培训所有的工程师，告诉他们在哪个点做什么事，项目发布之前要给安全工程师签字。

子柳：大学上的是少年班，在阿里巴巴成为最年轻的专家（23岁就是安全专家），在外人看来，你是一个天才，你怎么看自己的成长之路?

吴翰清：上少年班的事情，我认为最好还是不要拔苗助长，我在少年班的很多同学现在发展得并不是很好。那时候，周围都是年龄比较大的同学，谈恋爱都找不到对象，呵呵。我们那个年龄的学生都没有自制能力，到了学校之后就控制不了自己想去玩的那种心态，所以这样并不好。后来做安全完全出于兴趣，我觉得上少年班是为父母上的，而来阿里做安全确实是我自己想要的，能做到今天这个样子，也说不上有多大成就，主要还是因为我自己的兴趣。

子柳：在你的书中，你说过来阿里巴巴面试的时候，面试官让你展示一下技术，你把公司的办公内网给黑了，这个是怎么做到的？

吴翰清：这个不是公司的内网网站，是办公用的网络，当时我和几个朋友基本上把浙江这边的网络提前控制了，这边主要都是电信的网络。

子柳：给安全方面刚上路的技术人员一些建议吧？

吴翰清：从基本功做起，研究常见的漏洞，把它查出来，并去分析它，不要用它来做坏事。另外，去看看公开的漏洞，研究一下漏洞的利用技巧。

云铮——数据平台与产品部资深技术专家

张清，淘宝花名为云铮。2002年毕业于浙江大学计算机系，2003年初加入阿里巴巴，成功创建了阿里巴巴DW（数据仓库），经历了阿里DW从创建到成熟运用。2005年，淘宝数据开始启动，调任淘宝打造淘宝数据平台体系，推动商业智能体系建设。对国内外最新数据技术狂热，在和国内外业界行家交流及在阿里、淘宝的实战中，糅合所学所见，无门无派，注重实效，自成一套淘宝数据体系建设和运用的实战方法。八年数据生涯的梦想是能够推动集团数据统一体系架构，为数据化运营，推动实现十年数据战略尽自己一份力。

子柳：恭喜你当选大淘宝技术委员会数据分会的新任会长，今年数据分会有哪些工作计划?

云铮：今年（2012年）一个很重要的工作就是响应这个分会成员的呼声，加入这里到底是干什么的？写Job Model（任职能力模型）、参加晋升评审吗？不仅仅是这些，要让他们有引路人，有方向感。数据领域有很多模块，不能说让大家做了几年还局限在一个格子里，要把信息互通，把思维打开。

第二个就是数据技术的新趋势，这个是每年必做的，交流的范围也会越来越广，越来越深入。

第三个是把整个集团的数据打通。整个集团的数据团队有很多，B2B、支付宝、阿里云、阿里金融、淘宝都在做，这样力量比较分散，现在来看有必要把这些打通，形成合力，然后几个公司又各自有优势领域。如果我们拧成一股绳，阿里集团的数据在业界就没有人能够撼动。

子柳：淘宝网数据平台是你们一手带起来的，请讲述一下这个部门的发展历程。

云铮：我来的时候是一个数据分析部，那时侯有天宏、小龙女等几位同事做数据分析，我和正德几个工程师提供支持，捣腾数据库，建数据仓库。我们最初是先有业务，没有技术，根据业务需求做技术。一开始用一个单机的Oracle，然后经过一次组织

的调整之后，数据团队的几个工程师归七公带了，这个时候开始步入正轨，从单机的Oracle到4个节点、16个节点、20个节点。到20个节点的时候达到了Oracle集群的极限，这个时候已经是全球最大的Oracle集群了（Amazon美国是17个节点）。这时我们也发现商业的工具已经被我们用到了极限，接下来怎么办呢？我们只好自己革自己的命，把Oracle逐步替换掉，目前已完成了这个工作。

（子柳：这与在前台替换Oracle有什么不同吗？）这里面有很多复杂的业务，它的数据从前台拉过来。我们本身的环境在变，前台的环境也在变，新迁的环境是Hadoop平台，一开始我们对它也不是很熟悉。这里有三个变量同时在变，如果能控制一个不变的话，还好做一点，例如，等前台变完，我们再变，但这样业务不允许，我们对时效性的要求非常高。在“五彩石”项目中，前台上线的当天，我们跟着上线，必须保持步调一致。我们去找电信行业、金融行业、世界500强企业学习，发现他们的前台业务做变革时，后台数据的工作全停了，前台交付好后再做数据，“干嘛要同步做？本身就不应该嘛”。没办法，只能自己去想办法解决这样的问题，最后我们解决了这样的问题，这在业界是没人做过的，这是一个非常宝贵的经验。

随着这些创造性的工作的成功，整个团队也被锻炼出来了，再碰到什么新的问题不会害怕了，总有办法解决的。在发展的过

程中还有一个比较大的事件，就是阿里妈妈业务的合并。阿里妈妈是一个独立的公司，它从前台到后台有一整套的系统，前端要跟淘宝对接，后端所有的数据也都要整合。整合的细节很复杂，最终是成功整合了，前端形成了量子统计和数据魔方两个产品，后面数据的系统整合成Hadoop的一个集群。当时阿里妈妈有四五个集群，数据捣腾来捣腾去，其时效性、运维的成本都非常高。整合成一个集群以后统一管理、调度、维护、监控，慢慢地，报警越来越少，大家集中力量做业务、平台，这在当时是一个非常关键的工作，集中力量办了一件大事。

俗话说“分久必合，合久必分”。近两年，我们又孵化出了一些团队和产品，并进驻到了各个子公司。今年（2012年）又要在整个集团的范围内，和阿里云一起形成一个官方的组织，牵头促成“数据分享第一平台”模式创新工作，就是现在的“冰火鸟”项目。这里面汇集了集团里最大的两个数据系统——“飞天”和Hadoop，这个团队也包含了集团里做数据的几乎所有高P级工程师和核心架构师，是今年集中优势兵力协同做大事，相信这个项目能锻炼出更优秀、更具备大局观视角的一批数据技术专家。

子柳：数据平台经过这么多年的投入，目前产生了什么样的效益?

云铮：最简单的一个考量方式就是这些数据有没有变成价值，数据魔方和量子统计是把数据变成钱的两个产品，现在收入

还不错。但并不是说能赚到的钱就是它的价值，这些只是我们的一些尝试，我们才刚刚开始探索数据的价值，它未来是什么我们还不知道，它未来的价值更是不可考量。

子柳：我们的数据计算平台与Google、Amazon有什么异同？

云铮：从相同点看，这个级别的公司做数据，从宏观上看都是分布式的。Google做得早一点，他们自己开发的Bigtable、GFS，从分布式存储到分布式计算开发了一系列的产品，用在自己的搜索中。Amazon和Google又不太一样，Amazon采用虚拟机的方式，自己给别人搭OPS，用虚拟机租赁的方式做云计算，自己也有一些业务数据放在上面。

淘宝采用在开源的分布式平台上面用Patch的方式来做，从“云梯”到“飞天”这样的平台。分布式的理念是相通的，我们拥有全套“飞天”系统的自主知识产权，有不少精妙的设计，自主设计的后劲很足。

另外一个很重要的不同点是里面的数据是不同的，数据的价值也是不同的。淘宝把中国电子商务从零开始到现在，几乎所有的数据都包含了，有B2B、B2C和C2C的商品数据，以及交易数据和支付数据。而Amazon只有B2C的数据，Google没有商业数据，都是搜索的信息。淘宝的数据从量和质上面都非常高，而且更适合中国的国情，这上面是中国人的消费数据。这些数据的价值需

要持续地创新和在更大的生态链中去寻找和挖掘。

子柳：我们的数据现在达到了一个什么样的规模?

云铮：我们的数据每日新增长达到100TB左右，通过极限存储等创新的技术手段，控制净增量快速增长的势头。

子柳：作为一名互联网技术老鸟，你的成长之路是什么样的?

云铮：理想主义，兴趣+执著，看准一个方向后，无论是顺境还是逆境，都要不断地努力，不浪费时间和机会。

入职初期，我对数据非常有兴趣，然后就是马总在阿里巴巴成立初期就请了前亚马逊首席科学家Dr. Andreas Weigend给公司介绍了数据的价值和重要性。虽然当时没有完全理解，不过当时凭着一股冲劲，就持续做下来了。随着公司的发展，从B2B DW到淘宝数据平台，阿里云成立后，开始和阿里云一起看着飞天系统从梦想逐步变成现实，经历数据系统发展的这些宝贵经历，有机遇在里面，更需要持续的执著和努力，我庆幸的是我们有一批在这个环境里成长起来的核心数据技术人员。

子柳：从2003年加入阿里巴巴到现在，在技术岗位上做了9年（至2012年止），历经了多少次主动或被动的变迁？你怎么看待这些变动?

云铮：从DBA转型做DW，这个原因应该是公司开始启动搞数据时，没有合适的人，当时和数据最相关的应该是DBA这个岗位。因此，我有幸被抽到这个项目组，开始了Alibaba数据仓库的第一步建设。

第二次是从B2B到淘宝，应该是2005年下半年，这次是主动的，当时B2B的数据仓库已经投入使用，基本架构完成，业务也跑顺了，我也成为B2B数据仓库建设的核心成员，基本上很顺手了。当时的淘宝业务变化很大，从行业经验上讲，做数据仓库失败的可能性很大，而且难度高，于是决定来淘宝从无做起。

淘宝的数据发展的确很快，业务也在不断变化，不过这样的环境正好促使了数据平台的快速发展，随着淘宝前台几次大的重构、拆分、合并，数据基本上每年都会遇到以前完全没有遇到过的问题和挑战，但每次都顺利过关。随着淘宝五彩石项目的成功，以及广告数据平台和淘宝数据平台合并完成，业务开始迅速发展，成为数据应用平台的原始驱动力，这个时间段的技术和业务发展进入高峰期。

说到我对变动的看法，我一直是一个喜欢挑战的人，我认为有变动是好事，这会让人经历更多，而且应该主动创造变化，比如平台稳定了，系统理顺了，是不是就应该刀枪入库，马放南山

了？不是的，应该从更深、更全的角度去提出新的要求和新的梦想，并进一步去实现。

子柳：给技术刚起步的人员一些技术成长的建议吧。

云铮：兴趣是最好的老师，坚持是达到梦想的唯一途径，当然，在个人发展的不同阶段寻找到合适的导师很重要，看准方向会事半功倍。在刚刚参加工作还没有形成自己的判断时，方向有两个来源，一个是个人的兴趣，一个是找一个你非常佩服且能掌握未来方向的人，当然，如果这两者正好重合，那么剩下的就是脚踏实地坚持。

小马——淘宝UED前端通用平台高级技术专家

赵泽欣，花名为小马，2006年10月加入淘宝网，加入淘宝之前，基本上都在创业公司中度过，创业项目繁杂，角色兼顾产品开发、售前咨询、售后支持、客户培训。加入淘宝后，他成了全网第一位前端工程师，参与了旺铺、收藏夹、物流、Web旺旺和P4P等重要产品的前端开发，对淘宝整个交易流程做了系统的前端重构。先后主导并产出了最初的前端JavaScript类库TBra，组织虚拟团队优化淘宝各关键页面的前端性能，负责技术委员会速度小组的工作。2010年开始带领前端架构团队，负责前端通用框架的研发与推广，UDC 核心业务支持和前端质量保障体系的建设。

子柳：作为D2（前端技术论坛）的早期主要发起人，当时举办D2的初衷是什么？开始时举办的情况如何？

小马：当时的环境是这样的，前端的从业人员越来越多，行业也越来越趋于成熟，而这种属于前端从业人员的会议却很少，所以就想到是不是可以组织一场这样的活动让更多的同行聚在一起呢？所以D2 就诞生了。算上今年（2012年）的ADC 技术嘉年华，已经是D2 举办的第七届了，而回想第一届D2，当时我们的会场等条件其实都是相对简陋的，我还记得第一届的参加人数也就是70多人，发展到现在，D2 每次举办活动时参加的人数和业界的知名度都让我们感到很欣慰。

子柳：今年（2012年）的“ADC 技术嘉年华”是第二届，去年也是叫“淘宝技术嘉年华”，我们知道这个会议的主要组织者之一有你，能不能谈谈目前你对“技术嘉年华”的一些评价和你所期望的“技术嘉年华”未来可以发展成什么样子？

小马：其实说到咱们的嘉年华会议，我要说起一些国外的会议活动，比如我们参加了Velocity大会，当时令我印象很深刻的是，有一个嘉宾分享了一个观点：“性能更应该是功能……”，这让我想到很多项目其实一开始都很关注功能，当功能通过后，先草草上线，而后再逐步完善性能上的问题，这个嘉宾的说法让我产生了强烈的共鸣，这种体会在这个大会上有很多。因此，当参加完会议后，我们都是发自内心地感到受益了并佩服这种高质

量会议的举办方，当时我们也是想能不能把这种会议引进中国来举办，到现在我们也是“Velocity中国”大会的组织者。

说到我们自己的“技术嘉年华”，今年是第二届，就去年的第一届看，当时在那么短的时间和人力紧张的情况下，活动的组织方灵珊团队能够整合到这么多子品牌的力量，而会上的分享内容也丝毫不逊色于国内很多门票非常昂贵的会议内容，第一届的会议质量得到业界同仁的认同，这些完全都是超出预期的，因此，我觉得活动举办得很成功。

而对于第二届的期望，我们当时也是借公司给的机会去了趟美国，国外大会的展会展区给我留下了很多印象，比如，在国内很多会议的展台前就发放一些礼品、资料，有些就是放个盒子让参会者放入名片等，而在国外会议的展台前，我看到了更多的互动部分，他们会在展位前搞很多游戏式的互动来收集参会者的信息等，这种体验是很好的，让人觉得很舒服。所以，我当时体验了每一个展台（笑）。

我想我们国内的大会除了分享高质量的内容外，是不是也能在大会现场多一些这种轻松的互动元素？希望我们的“技术嘉年华”活动让更多的技术人员聚在一起分享到高质量内容的同时，是不是能够让他们感受到更多技术带来的快乐？本届“技术嘉年华”也是租下了一层的空间布展，一种新的尝试，希望届时的活动能让大家真正放下工作上的压力，得到轻松的体验，这样由

“精彩的主题”+“轻松快乐的气氛”组成的会议，才是名副其实的“技术嘉年华”。

对于“ADC技术嘉年华”的未来，我希望我们能够继续保持特色，坚持最初的想法，在节约公司资源的同时继续把握非商业性会议的方向，为阿里的子公司和依赖于阿里生态圈生存的企业技术人员提供一个平台，通过这个平台可以更加及时全面地了解阿里，我们的会议成为业界风向标是基础，我们更要向业界的F8会议、Google I/O看齐。

子柳：小马是淘宝网第一个做前端的，目前还在做前端，现在有5年多了，给大家简单介绍一下淘宝前端的发展历程吧。

小马：先给大家说一下“前端”是怎么来的，我自己对前端有一些思考。就以淘宝为例，一开始，淘宝就是一个社区，大家发布信息给别人看。随着互联网的发展，大家不仅仅要看到一个静态的页面，他们更需要有绚丽的页面，有更丰富的交互。也就是说，对“体验”的要求越来越高。这样交互难免变得越来越复杂，这个时候开发人员并不是很适合做这些事情，而是需要一个细化的工种来做这件事情。我个人认为传统的开发人员更多的是面向机器来开发的，更多的是考虑CPU的问题、内存的问题、数据库的问题。实际上需要一些开发人员对“用户”有更敏锐的感觉，他们来完成“界面”的问题，界面是互联网与用户交互的地方，他需要对体验有一些认识，对“人”更关注。从另一方面来

看，这几年下来，客户端越来越多，页面要适应很多客户端，这也不是传统的开发人员擅长做的事情。基于这些原因，就产生了这么一个新的工种。

进入淘宝网后，一开始要我做的就是“怎么让淘宝网的页面动起来”，让页面展示更丰富的信息，让用户的体验更加流畅。当时还没有前端工程师这个职位，我入职的职位是“Java工程师”。 淘宝前端的工作就这样开始做了，我们刚开始做的工作就是让页面动起来，做了很多JavaScript开发工作，让整个页面变得更加活泼，这是第一阶段淘宝前端的状况。

到第二阶段，随着淘宝的业务变得越来越复杂，页面也变得越来越多，前端开发的工作量就变得越来越大，这时就需要一个很大的团队来做这个工作。与此同时，“AJAX”变得流行起来，它改变了整个业界的开发模式，很多工作要通过前端来实现，这也让前端的工作越来越多。这个阶段前端的团队迅速成长，你可以理解为，第一阶段就是让页面动起来，第二阶段是前端和后端合作，让页面的体验更加友好。

再往下一个阶段，我们发现越来越多的内容在浏览器中展现，越来越多的浏览器出现了（Firefox、Chrome、IE 6～IE 8），这让前端越来越重，就出现了性能的问题，我们又需要做一些框架、规范和很多优化的工作。

到现在这两年来看，互联网经过这么多年的发展，开发模式渐渐变得比较稳定和成熟了。传统的开发人员不擅长的事情现在有了强大的框架，方便他们很快上手使用。从企业的成本看，原来需要两个工种去做的事情，现在一个工种就可以完成。这样慢慢出现了一个趋势，貌似不太需要把工种划分得这么明细，以后对淘宝来说也会继续存在前端这个工种，但更多的是做一些架构和统筹方面的工作。

子柳：现在很多团队都在经历着分分合合的变动，从前端团队来看，是很典型的一个例子，你怎么看待这些变动?

小马：刚开始，我们的团队就四五个人，大家在一起按需做页面。随着网站业务的拆分，开始有一些分工，例如，有人负责商城的前端开发，有人负责主站的前端开发，虽然业务上有一些分工，但团队还是一个。再往后面，随着淘宝主站的拆分，就出现了“业务线”的概念，例如，商品线、店铺线等，这个时候前端团队进行了一次拆分，分成几个小团队进入每个业务线。这样拆分有利有弊，好处是能快速推动业务的发展，弊端是同一工种之间缺少交流，会产生很多重复建设。分久必合，过了一段时间，团队又合并到一起了。合久必分，没多久，整个公司也拆分了，分成几个子公司，我们当然又得拆，团队分别拆分到几个子公司。我觉得合和分的过程也不用特别看重，不同的阶段需要不同的组织形式。

到现在这个阶段，应该说是半分半合，为什么说是半分半合？因为前端团队分成了两块，一块跟进业务方面的前端开发，一块做通用产品的开发，这样保证了业务能够得到有效的支持。另外，对于通用的产品，能够有效地管理起来，避免了大家开发环境的不一致、标准的不统一等问题。

子柳：畅想一下前端未来的发展方向。

小马：我认为，未来的方向是前后端的界线越来越模糊，即未来求开发工程师能够把前后端的工作衔接起来。我个人比较认可一种说法：经过这几年开发模式的渐渐发展，未来前后端的开发会融合起来，这样的岗位叫做“Web开发工程师”。

另外，这个岗位叫什么名字不重要，重要的是人们对于交互和体验的要求越来越高，随着移动互联网的应用，会产生更多的交互方式，如滑动和震动等。这就要求有人做这方面的开发工作，同时还要推动前端技术的发展。

子柳：单从技术方向上看，HTML5是不是未来的一个大趋势?

小马：HTML5不是“HTML4”的简单升级，很多人会认为HTML5就是那些标签括号括起来的表示性的语言，其实HTML5和HTML4不是一回事儿，我们现在所说的HTML5除了有HTML4增加的一些语义性的标签之外，通常，我们把CSS3和很多新的

JavaScript（简称JS）的API都合起来统称为HTML5。为什么会有这些变化？我觉得原来的HTML4和一些相关的技术并不能让Web成为一个很好的开发平台，它只能让Web作为一个“界面”展现一些内容，做一些简单的交互。而HTML5的目的是想让整个Web真正成为一个开发平台，或者说是让浏览器成为一个适合开发大型应用的平台。你看它的变化，首先HTML5标签的变化是让它更具有语义化，然后CSS3把很多展现型的东西做了加强，最重要的是新增的那些JS的API，你现在已经可以在浏览器本地连接数据库、使用Socket、使用本地存储、获取地理位置等，很多我们以前开发过程中需要从后台取得的信息，现在都可以在本地浏览器中做。这使得浏览器变成一个适合做大型应用的平台，而不是像以前一样只做内容的展示。

在淘宝的网站上，HTML5已经无处不在，淘宝是国内应用HTML5最早的网站之一。

子柳：在发展的过程中，我们从业界获取了很多支持，描述一下业界对我们的影响有哪些？

小马：有三家公司对我们影响非常大。

第一家是Yahoo（雅虎），“前端”岗位的定义就起源于雅虎。当时雅虎的发展如日中天，他们有很多优秀的技术和人才，并且最早提出前端的概念、框架和规范。我们最早使用的JS框架

YUI就是雅虎创造的，我们的很多规范也是借鉴他们的，我们曾遇到的性能问题就参照雅虎提供的优化方案和工具（YSlow）。可以说，2007年至2009年，我们很多东西都是借鉴雅虎的。

第二家公司是Google，Google为什么对前端产生这么大的影响呢？是因为AJAX实际上是被Google变成一个商业上很成功的应用，它的Gmail、Docs等产品采用了大量的AJAX技术，AJAX在Google的成功应用让很多前端和交互的工程师开始使用这种技术，让前端的工作变得更加繁荣。

第三家就是Facebook，Facebook是把体验和技术结合得最完美的公司。

其实看起来对我们影响最大的也就是硅谷历史上走在最前端的上述三家公司。

子柳：这三家公司有两家都被“墙”了（即被屏蔽，无法打开其网站），很多人知道“墙”阻碍了信息的交流，却不知道“墙”对技术的发展影响很大，“翻墙”是技术人员必备的技能。

小马：确实是的，最搞笑的是有个日本女优的名字就叫YUI，有段时间搜索YUI，就会被屏蔽掉。

子柳：那回过头来说我们对业界的影响有哪些呢？

小马：我们通过博客、主办的D2和翻译的一些图书比较成功地把“前端”岗位推到了业界，让很多公司知道这个工作的重要性，也让很多技术人员知道自己从事的是前端的工作。

子柳：有了孩子之后，对你有什么影响?

小马：影响很大。其实不是从有了孩子开始，从知道老婆怀孕就开始了。以前，我的工作时间相对比较自由，时间比较充足，从老婆怀孕开始就要给她做饭，有了孩子又要陪孩子，时间变得非常碎片化，必须更有效地利用自己的时间。另外，责任感有了明显的提升，例如，我以前开车就很少系安全带，有了孩子之后就不愿意冒任何风险了；以前感冒了就耗着，现在怕生病传染给孩子，就开始关心和爱惜自己的身体；在孩子面前也不敢说脏话了。其实孩子的成长也督促着我自己的成长。

子柳：你和你的团队都翻译过好几本书了，有什么经验给大家传播一下?

小马：翻译的都是有关前端方面的书，一开始是因为觉得我们需要这方面的技术，这也可以推动前端技术在国内的发展。写书也不像写博客那么随意，这是要变成铅字的，所以要很谨慎，这就要求必须把后面的原理弄明白，对于已经掌握的东西也相当于复习了一遍，这是一个很好的学习过程。关于团队一起写书，

我发现这是一个很好的团队建设的方式，在写书的过程中，大家要有很多交流，能够互相学习，互相督促，也增进了彼此的感情。推荐其他同事也试试看。

另外一点经验是写书是不赚钱的，基本上可以说是公益性质的，这要占用大量的时间。

子柳：说说你个人的成长经验，给起步阶段的同事一些建议吧。

小马：现在回过头来看，其实成长最快的一段时间是刚进淘宝的那几年，那个时候很单纯，就想着把工作做好，做完一个做下一个，不管这个业务是不是重要，需求方是不是好打交道。有个“一万个小时理论”，我觉得很正确，说的就是一个人必须经过不断地练习，不断地遇到问题才能成长起来。当然，做的时候要不断总结，写博客是一个很好的途径。

子柳：现在有很多人出去创业，作为一个曾经创过业的技术人员，你有什么看法？

小马：很多人创业是因为公司变得越来越大，有很多流程和规章制度的限制，这样自己发挥的空间越来越小。但是真的到社会中创业时会发现限制同样很多，你同样要面临税务局、工商局、商标、公司注册、股东关系、员工管理、同行竞争等问题，

这跟你在公司中工作并没有本质的不同。所以说创业其实是一种心态，你用旺盛的精力、饱满的斗志和坚定的信念去克服一个又一个的困难，这就是创业。到外面去创业和在一个大公司里创业，我个人觉得并没有本质的区别。

当然，如果你不认可公司整体的方向，或者你不认可这个行业，你有一个理想去打造一个新的天地，公司也应该有这种胸怀鼓励你去创业。

淘宝传奇工程师多隆的程序世界

多隆是淘宝的创始人之一，也是淘宝的第一个程序员，他奠定了诸多淘宝重大软件项目的基础。有人说他是淘宝的“扫地僧”，有人说他是“神”。在淘宝，他做到了既懂C/C++语言，又懂Java和内核；既可以深入技术底层，又能切入到高层业务领域，从前端到后端，知识既广又深。他就是核心系统部专家组的多隆。

技术小二中流传一句话——“有困难，找多隆”。关于这点，我深有体会，有一次，我们组解决一个Apache服务器无故崩溃的诡异问题，搞了三天还没找出原因，于是请教多隆，他在三分钟后就告诉了我答案。瞬间的秒杀，让我领教了“高级研究员”的威力。

我和多隆在同一个部门，工位相邻。这个近水楼台先得月的条件，让我平时有机会观察他，从他的一举一动中思索他如何以非科班出身（生物系生命科学专业）成长为计算机牛人。

多隆说他知识经验的积累主要归功于在淘宝业务发展的过程中，他遇到了各种各样的问题。这些问题促使他不断学习解决问题的各种技术，他和淘宝一起成长。在我看来，他对技术始终保持着谦卑的心态也很关键。他把自己当成海绵一样去吸收新知识——在他的字典里，没有不值得去解决的问题，也没有不值得

去学习的技术。而且每学一个知识点，多隆都会写一段代码去验证，一方面是练习，另一方面也让他加深理解，直到真正掌握这个技术。

多隆还有一个常人难以做到的特质。当他沉浸在他的程序世界时，外界的人和事很难干扰到他。一天的工作时间里，他绝大部分都在座位上写代码。若他不在座位上，那基本上就在洗手间。我还记得2010年公司的乒乓球比赛决赛是在创业10楼的休闲吧举行的，比赛现场距离多隆只有20米远，锣鼓喧天，人声鼎沸，很多人都被吸引过去了，整个办公区只有多隆一个人还“粘”在椅子上。这大概就是《功夫熊猫2》里的最高武功心法“Inner Peace”（内心 平静）吧。有了这样的专注力，不成为高手也难。

多隆从2000年加入阿里巴巴，到现在已经十多年了，仍在淘宝技术第一线写代码。我曾问他是如何坚持这么久且至今还这么有激情。他回答说，很简单，因为他在做他喜欢的事情，解决问题和写代码让他觉得很有成就感。有一次，我们在从庐山郊游回来的火车上，他还在写代码，可见他对写代码喜欢到了什么程度。

在淘宝，多隆被从副总裁到普通的软件工程师等诸多同事衷心佩服，广受爱戴。这里面既有大家对多隆技术上的认可，更有大家对他默默付出和人格魅力的赞叹。2011年的公司年会上，那个《淘宝的一年，亲》的视频曾感动了很多小二。很多工程师为了淘宝线上的稳定，奉献了很多，也对亲人和家庭亏欠了很

多。看视频的时候，坐在我旁边的多隆泪流满面。我想，他之所以有如此深的感触，是因为视频里的故事正是他无数个类似不眠之夜的写照——2009年之前，公司甚至还没有视频里提到的“消防群”这个集体解决、承担故障的组织，即使是半夜，多隆都要起来解决问题、排除故障。而求救过他的小二说他总是随叫随到，没有架子，态度和蔼，任劳任怨。直到现在，很多消防群里解决不了的问题，仍然会找多隆来解决，而他也会在第一时间出现。

一个计算机工程师该以怎样的态度和方式来工作和学习？多隆的一条朴素的建议或许可以很好地解答：“发现问题，解决问题，不要绕开问题的本身；多做事情，不会吃亏，即使不是你的事情。”这大概也是多隆最大的成功秘诀吧。看似容易的原则，却不是每个人都能做到的。做到了，你也有希望成为“多隆”。

始终保持对代码的那份单纯的热爱，保持对技术的专注和钻研；别人把工作当工作，他把工作当事业——这就是多隆的程序世界。

（注：本篇作者为叔度）