适合开发者 项目经理 CTO&CIO 编程爱好者阅读收藏

Broadview®
www.broadview.com.cn

PROGRAMMER 程序员

PROGRAMMER 程序员 2014 精华本

程序员编辑部 编

7大篇章，12期精华

讲述成功产品背后的 技术、人和事

专题篇 ■ 对话篇 ■ 管理篇 ■ 产品篇
移动篇 ■ 技术篇 ■ 云计算篇

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

程序员网址：http://programmer.csdn.net

程序员2014精华本

程序员编辑部　编

总策划：孟迎霞

编委会：蒋　涛　刘　江　董世晓　杨　爽　卢鹄翔
徐威龙　蒲　婧　张　勇　万方宜　纪明超

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

程序员 2014 精华本

图书在版编目（CIP）数据

程序员 2014 精华本 / 程序员编辑部编. —北京：电子工业出版社，2015.2
ISBN 978-7-121-25471-0

Ⅰ. ①程… Ⅱ. ①程… Ⅲ. ①程序设计－文集 Ⅳ. ①TP311.1-53

中国版本图书馆 CIP 数据核字（2015）第 022758 号

责任编辑：董　英
印　　刷：北京京科印刷有限公司
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：850×1168　1/16　　印张：29.25　字数：1363 千字
版　　次：2015 年 2 月第 1 版
印　　次：2015 年 2 月第 1 次印刷
印　　数：5 000 册　　定价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

时光荏苒，每年一度的《程序员精华本》又和读者朋友们见面了。《程序员 2014 精华本》紧紧围绕大数据、电商架构、智能硬件、移动开发、团队管理等热门话题，进行了全面而深入的解读。

基于原有栏目和本年度热点，《程序员 2014 精华本》的结构分为以下七个篇章。

专题篇：综合了 2014 年 1-12 月封面报道，内容包括双 11 背后的技术、大数据实时处理、编程语言、我们是 MAKER、云计算理想与现实、移动开发工具、打造高效团队、安全新知、微信开发、移动 5 年、电商峰值系统架构设计、2014 TOP50 最具价值 CTO 等 12 个主题。

对话篇：由程序员记者直接采访大师级人物和知名技术人，在思想、实践等方面进行深刻碰撞。

管理篇：主要是来自软件方法、个人成长、一分钟先生等栏目的真知灼见。

产品篇：主要分享产品设计方面的方法、理念和实践。

移动篇：汇聚移动开发领域的观点、工具和技术。

云计算篇：云计算和大数据领域的热门技术和实践，特别是 2014 年大红大紫的 Docker 和 Spark。

技术篇：包括语言与工具、技术实践等方面内容。

希望您能喜欢《程序员 2014 精华本》。顺祝 2015 年顺利、安康！

程序员编辑部

2015 年 1 月

目 录

专题篇

对话篇

管理篇

产品篇

移动篇

云计算篇

技术篇

双 11 背后的技术

2013年是阿里双11的第五个年头，在双11当天天猫和淘宝单日成交350.19亿元人民币，网站PV过百亿。支付宝实现成功支付1.88亿笔交易，再次刷新了去年同期1亿零580万笔的全球纪录，最高每分钟支付79万笔。

这一系列数字令整个业界倾慕不已……

然而，在疯狂的业务数据背后，无疑是对阿里集团技术团队的一次整体大阅兵，从天猫、淘宝到支付宝，从PC到无线，从研发到运维保障，各个环节都面临着巨大的挑战。

本期封面报道以“双11背后的技术”为主题，力邀来自天猫、支付宝、中间件、运维保障等部门的一线技术人员，与大家一起解析其中的技术挑战与解决方案。

感谢2011–2013年双11天猫技术负责人庄卓然对本期封面报道的大力支持，以及各位作者的积极参与。

网购狂欢节背后的技术阅兵

文 / 庄卓然

从光棍节到网购狂欢节

2013年是双11的第五个年头，从2009年的“光棍节”到现在的“网购狂欢节”，单单是名字上的变化，大家就能从中感受到业务规模发生的转变。

2013年双11，天猫和淘宝单日成交额达到350.19亿元，网站PV过百亿，成就了14个销售过亿的商家。其中，有76%的商家处理工作是在聚石塔云计算平台上完成的，并且无一漏单、无一故障。

支付宝实现成功支付1.88亿笔交易，再次刷新了2012年同期1.0580亿笔交易的全球纪录，最高每分钟支付79万笔交易。

手机淘宝整体支付宝成交额为53.5亿元，是2012年的5.6倍（9.6亿元）；单日活跃用户达1.27亿；手机淘宝单日交易成交笔数达3590万笔，交易笔数占整体的21%。

天猫双11共产生快递包裹1.52亿件，仅双11当天，全国各大快递企业共处理6000多万件包裹，是2012年双11最高峰的1.7倍。

在疯狂的业务数据背后，是阿里技术团队的一次整体大阅兵，从天猫、淘宝到支付宝，从PC到无线，从研发到运维保障，各个环节都面临着巨大的挑战。

双 11 业务生态

在开始介绍这些挑战之前，我们先用一张图来看一看，作为消费者，你在2013年的双11大促中都会经历哪些业务过程，如图1所示。

首先，你会通过各种终端（PC、手机、平板）访问淘宝、天猫、聚划算的导购市场，丰富多样的导购产品会帮助你发现和挑选自己心仪的商品。大数据的基础平台支撑了大量有价值的数据应用，以保证你在这个环节的购物体验。稳定的交易平台确保了优惠价格的准确计算、库存的及时扣减和担保交易的订单有效生成。这时，你就可以通过支付宝安全放心地进行支付宝余额或者网上银行支付，满心欢喜地等待包裹的送达。

这时就开始轮到面向商家的系统忙碌起来了。聚石塔提供的云推送功能在第一时间将交易订单同步到部署在聚石塔云工作平台上的商家ERP、WMS、CRM软件中，并且为这部分软件提供了动态弹性扩容的能力和安全方面的有效保护，以便大商家在大量订单面前，可以像日常情况一样，快速组织商品出库和发货，而不用额外担心自己软件的处理能力。

最后出场的是菜鸟网络打造的雷达预警系统，通过大数据的力量，对商家的备货、消费者购买行为、物流服务能力进行预测，并与国家气象局、交通部实时发布的天气、道路情况进行同步运算，将未来半天至七天的预测结果反馈到13家快递公司，以便快递公司提前调配运力。最终，所有的包裹得以第一时间送到消费者的手中。

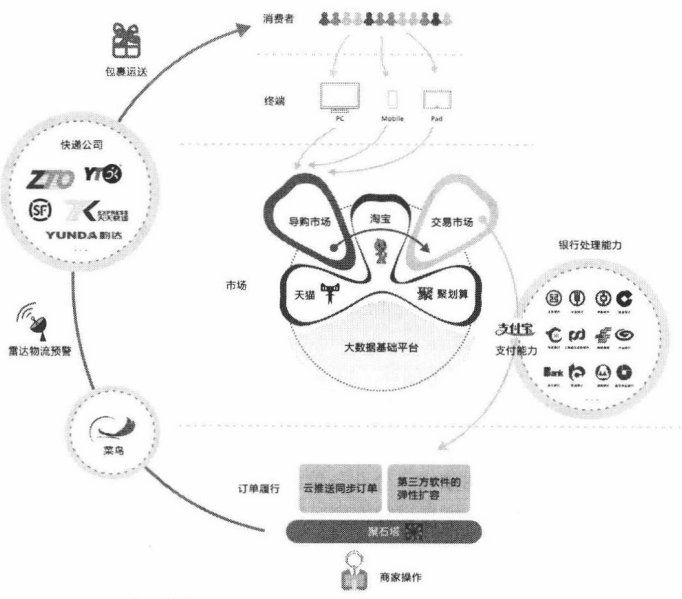

图 1 双 11 大促时的业务过程

挑战和技术准备

业务跨数量级的快速发展对整体架构带来了新挑战

经历了过去几年“去IOE”（IBM小型机、Oracle、EMC高端存储）整体架构改造，整个阿里集团建立了基于中间件、PC Server的轻量级分布式SOA架构，保证了服务器、数据库、网络、机房各个层面无单点，可以以较小的成本支持水平扩展，提升网站的负载能力。但随着这几年双11业务的飞速增长，PV、同时在线用户数、峰值交易和峰值支付等核心指标都开始跨越到新的数量级，架构在更高负载能力要求下的一些缺陷开始暴露出来。

第一，大规模访问请求带来的机房网络瓶颈。

双11当天有数亿用户访问天猫和淘宝的系统，高峰期甚至有数千万人同时在线。以天猫的“商品详情”页面为例，集群峰值QPS达到了十万以上的量级，是平时峰值QPS的数十倍。这样的突发性流量增长，使得机房的网络容量面临着巨大压力。如何能够利用合理的成本应对瞬间飙高所带来的网络压力，确保活动完整周期内用户响应时间的稳定性，以及局部出现问题时的高可用性，成了首先需要面对的问题。

着眼于未来，从2011年起，我们就开始了对浏览型的应用进行新的架构改造。历经页面细粒度的动静分离、统一缓存接入、CDN静态化三个阶段。最终，将更新频率不高的内容伪静态化直接缓存到CDN，通过统一的秒级失效机制控制缓存的更新操作，让绝大多数内容请求不需要回流到主机房，直接在用户最近的CDN节点就能够完成。这种方式一方面极大地缓解了主机房网络的压力，另一方面也优化了用户页面的响应时间。徐昭的《天猫浏览型应用的CDN静态化架构演变》将带大家更深入地了解我们是如何经历这一过程的。

第二，超大规模系统如何继续保持在不同层面上的水平伸缩性。

首先，服务器需求的增多，导致未来单一地区的IDC资源已无法满足容量增长的需求，机房供电能力成为短期内无法逾越的瓶颈。其次，简单地横向扩展IDC机房，机房之间的网络流量又成为了新的瓶颈。多机房的应用、缓存、数据库等访问的相互穿透，也加剧着跨机房的流量增长。最后，核心服务集群的应用服务器的规模增长，使对应的数据库连接数成为了瓶颈。每增加一台数据库服务器，对应的应用服务器都需要连接上来，数据库的连接数马上就不够分配了。也就是说，在今天的业务规模下，我们无法继续针对核心服务的数据库层再做横向的水平扩展了。

为了解决以上问题，2013年双11我们尝试了新的逻辑机房的架构方案，将数据水平拆分（Sharding）的思路向上提到接入层。以支付宝为例，先将完成某一特定业务需要的系统、核心服务、数据库组合抽象成一个业务单元（Zone）的概念，业务单元从应用服务器、缓存到数据库可以独立封闭运行。然后，从入口处根据用户请求路由到不同的逻辑业务单元，实现了单元级别的可伸缩性。不再依赖同城IDC，让交易、支付这样复杂的业务单元具备了大规模跨地域部署的能力。胡喜的《支付宝的架构变迁》会有更加深入和全面的介绍。

第三，如何更大程度地“压榨”单服务器的系统资源。

虚拟化技术已作为各大网站提升物理机资源利用率的基础技术。以天猫和淘宝网为例，2010年我们引入Xen虚拟化技术，1台物理机装3个虚拟机，一定程度上降低了成本。但随着机器规模的快速增长，我们发现其中1/3的虚拟机的Peak load<0.5，这意味着我们的运维成本还是不够低。主要有以下几方面原因：

- 单台物理机上跑的应用不够多；
- 分给应用的机型及机器数是静态的；
- 集群的资源利用率不均衡。

为了最大化物理机的资源性能，从2012年开始，我们大规模地应用了新的基于LXC的虚拟化方案，成功地在每台物理机（16Core/48GB）运行了12个应用实例，物理机的load提升到2~10。这对大促时期的成本控制非常有帮助，也为内部私有云的完整构建，摸索了一条可行的路径。

消费者对用户体验有了更高的要求

千人千面的购物体验

有一个很现实的问题摆在了我们面前：大促当天有几亿消费者会来到我们的网站，在上百万商家和过亿商品里面挑选自己心仪的宝贝。光靠运营人肉制作的活动页面和消费者的主动搜索已远远不可能满足需求。因此，需要在产品上转变思路，营造从以业务为中心到以用户为中心的大促体验。双11应该是面向消费者的“我的双11”，而不是“天猫的双11”。

基于此，2013年的双11大促中大面积应用了个性化算法，从PC到无线，从“会场”到“详情”，真正意义上为消费者打造了一个“我的双11”。从最终效果来看，2013年双11的用户购买转化率提高了10个百分点。张奇会在《个性化技术在双11的应用》中介绍在个性化推荐系统架构、机器学习算法应用和算法的快速评测方面的思考。

多终端的业务一致性

移动智能终端的快速崛起，让更多的消费者可以随时随地访问天猫和淘宝，但也让我们开始深入地思考，在业务快速发展的前提下，如何在不同终端上快速提供本质相同的服务。2013年我们首次在预热期的大型活动中尝试了基于Canvas的Web App解决方案，极大程度地提升了开发效率。天猫前端团队的鄢学鹍会在《双11的前端实践》一文中分享Mobile First的理念是如何在双11中实践的。

稳定性的极致要求

容量预估、依赖治理、监控

这一环节是历年双11技术成功与否的关键环节。中间件团队的《中间件技术的双11实践》除了会介绍在SOA架构体系中扮演重要角色的中间件产品之外，也会就容量预估、依赖治理和监控的双11实践做精彩介绍。

业务降级、限流预案

从2010年双11开始，每年都会有针对性地准备一些技术预案，在流量超出预期时做好限流保护，在局部系统处理能力出现瓶颈时进行优雅降级，以提升整体的吞吐率，保证核心业务的正常运行。

2013年，我们在技术环节准备了2000多套应急预案，大到

遭受骇客攻击、各类业务应急动作，小到服务器机房空调发生故障，均有详细预案。不仅各服务器机房，甚至西溪园区双 11 大促指挥办公现场都准备了柴油发电机。此外，2013 年针对双 11 进行了上百次的内部演习，其中全网大规模演习就进行了 10 余次，以确保每一个预案的有效性。

全链路压测

压力测试对于评估网站性能的重要性是不言而喻的，但无论是线下模拟的单一集群压测，还是线上引流压测，都只能暴露一些基本的单点问题。对于双 11 当天高峰期的真实压力模拟，这两种传统的压力测试方式还存在着巨大偏差。

首先是业务处理链路的复杂性，对于像天猫这样一个分布式处理平台，一笔交易的创建会涉及多个应用集群的处理，在能力评估时也应该考虑的是一个处理链路而不仅仅是单一应用集群的处理能力。其次是应用之外的风险点，像网络、数据库等，很难在传统压测中体现出来。

为了精确评估核心交易集群中各个环节的能力瓶颈，2013 年我们实现了一套新的全链路压测评估体系。通过线上真实用户数据与人为测试数据相结合的方式，首次成功地在生产环境中模拟出相对真实的超大规模访问流量，将前端系统、网络、数据库等一整套系统环境完整地纳入压测范围，贴近实际的应用场景，为评估淘宝和天猫交易核心链路的实际承载能力提供有说服力的数据依据。这一方面可以验证交易核心链路上各种限流和预案的准确性，另一方面也可以充分暴露全链路上的各种瓶颈和隐藏的风险点，让压力测试的工作真正落实到了确定性的层面上。

基础运维能力提升的预研

在支撑好现有业务的同时，如何将基础运维能力（高稳定性、弹性调度、低能耗、快速交付等）提升到新的量级，成了阿里技术团队未来面临的新挑战。为了迎接这方面的挑战，2013 年双 11 我们也小范围做了一些面向未来的基础运维预研工作。

首先，可以预见的是，随着业务的不断发展，阿里会有越来越多 IDC 布点。虽然逻辑机房的方案已能解决交易支付环节场景下的跨机房调用问题，但随着云计算的推广，我们依然面临着大量跨机房调用的场景。现阶段，交换机的网络路由主要是基于 IP 做简单识别，并不会通过识别应用类型提供智能调度。当大规模出现多个 IDC 时，跨机房的用户体验问题会越来越成为瓶颈。为了解决这一问题，我们 2013 年在交换机的 OS 层面做了大量自主研发工作，定制自主的交换机 OS，实现软件定义网络（SDN），对管控和成本两个层面都有较大的改进。庞俊英的《阿里的 SDN 实践》将简单介绍我们如何让应用流量通过 SDN 识别出来，从而实现不同流量的长途链路和短途链路调度。

其次，在未来的 3~5 年，阿里的服务器将扩大到几十万甚至百万级的规模，如果延续以前按服务器进行交付的模式，交付速度和能耗都会很快出现瓶颈。因此，我们 2013 年尝试了整机架服务器解决方案，通过标准化、流程化、定制化，解耦各个交付环节在 IDC 现场部署的时间串行关系，提升交付的质量和效率。除此之外，其中例如统一机架 Backbone 模块这样的设计，将整体的耗电量降低了接近 10%。放在百万级服务器的规模下，无疑会节省一笔极大的开销。武鹏的《阿里整机架服务器解决方案》会展示我们在低能耗、快速交付上的思考和方案。

经验小结

对于技术人员而言，双 11 绝对是一次奇幻体验，在这里有丰富的场景可以实现个人技术上的奇思妙想，同时也经历着最极致的技术考验。经历了 5 年双 11，我来小结一下技术准备上的一些经验心得。

提前明确架构目标。架构和基础技术的调整往往带来应用系统的伤筋动骨，前面介绍的一些大架构改造，周期往往需要 2~3 年，试错成本非常高。因此对于架构师和技术核心决策者而言，要有足够的前瞻性，在选型设计上一定从全局发展和核心问题出发，准确定位、明确目标，才能统一团队的整体方向。

不过度追求过程中的完美。我们在内部总把系统架构的过程类比为建造水坝——上下游级联，同时兼顾对于全局生态的影响。但落实到实施层面才是挑战的开始，大家总是会倾向于考虑是否优雅完美，从而让方案的复杂度不断提升。架构师需要能够兼顾成本、资源、时间等各方面的因素，敢于做取舍，勇于舍弃一些收效甚微的优化。到准备后期更是如此，发生问题不可怕，可怕的是解决问题的时候又引入新问题。没有 100% 完美的方案，能满足业务现阶段发展需要的方案就是好方案。

细节决定成败。分布式系统的好处在于松耦合、灵活性和可快速扩展，但同时也带来了一系列麻烦，包括数据的一致性、分布式事务、各种应用在服务层的数据相互干扰等。在大促这样的峰值压力下，这些小问题都会被无限放大。因此，前期准备过程中要谨记不能放过任何微小的细节，侥幸心理更是万万要不得，担心出问题的角落往往一定会出问题，最没问题的地方或许会是风险最大的地方。另外，尽可能工具化一切人为操作环节，在架构上做好约束，同时时刻确保留有“Plan B”，只有这样才能够确保对最终结果有把握。

最后，350 亿绝对不是终点，数字本身并无特殊意义，也不会是我们的终极需求，阿里“让天下没有难做的生意”的使命还在继续。用技术的力量重构整个电商生态，改善更多人的生活，还有很多路要走，让我们一起共同期待更多的技术奇迹。

双 11 的前端实践

文 / 鄢学鹃

每年双 11 的第 1 秒和第 1 小时都是并发量和流量最大的时刻，安然度过这 1 秒和这 1 小时，从技术角度讲就意味着双 11 成功了一半。按照每年双 11 的规模都有 2~3 倍增长的规律，我们就可以大致估算出 2013 年所面对的峰值，这是所有技术方案面临的挑战底线。前端团队担负全站人机交互界面的实现和品质保证，每年双 11 都会遇到如下挑战。

- 短时间内完成巨大的界面开发量（天猫全站各种双 11 特有的界面表达和数百个主 / 分会场），并精确到秒地将其发布，

同时要快速响应界面调整，实现全站实时发布。

■ 大并发量和大流量面前，为了减轻后端服务的压力和提升前端性能，需要对页面的部分内容进行分级 / 降级（不同的情况下降级不同的内容）。这需要小心设计业务的模块，部分前端代码的临时下线不影响整体，或者服务端压力过大对流量进行限制时，前端如何对用户进行友好的显示和持续的引导。

■ 数百个运营活动涉及非常多的内容维护和策略调整，如何保证不出现错误的图片、错误的链接以及性能保持一致高效。

除了这些传统挑战，2013 年双 11 的用户环境发生了巨大变化，用户终端的碎片化导致现在不仅面临多个终端（桌面、平板、手机）多个浏览器的兼容，还要应对终端和终端之间的互动。下面将从五个方面来谈谈我们应对这些挑战的思路、方案和经验。

跨终端

移动智能终端的兴起，用户不再像以前一样只是停留在桌面上，平板、手机的大量涌现使得用户终端碎片化。对电商平台而言，直接导致了如下问题。

■ 导购渠道碎片化：消费者在各个导购渠道上看到的核心业务不一致，相同业务的核心人机交互和业务逻辑差异较大。如图 1 所示。

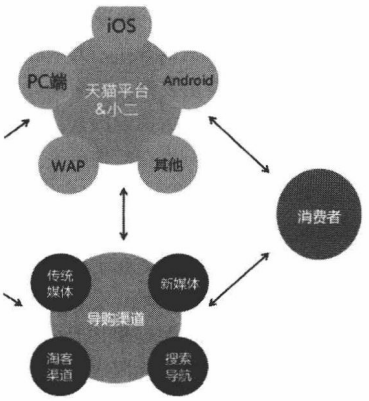

图 1 导购渠道碎片化

■ 商家和小二各导购渠道的运营需求无法满足。

■ 实施成本高，基础设施重复建设。

针对这些问题，我们得出以下结论。

■ 用户流和信息流在跨终端流动，它穿越了传统终端（电视、广播、平面媒体等）、智能终端（手机、平板、桌面、电视等）和软件终端（浏览器、社交网络、即时聊天软件和其他应用），这些流动不是一维单向平面的，而是多维双向立体的。

■ 移动用户正在快速成长，正在或即将成为“大多数”，同时移动智能设备的人机交互形式代表未来，会引导桌面的人机交互形式向移动化变革。

■ 人的本性、业务的本质和商业模式的本质不会随着终端的改变而改变，只是形式会随着终端而进化。

因此，为各种碎片场景用户提供本质相同的电商服务，技术上面临最大的挑战就是如何快速、高质量地开发跨终端产品。如图 2 所示，我们的核心思想是利用 Mobile First 的理念，同一个团队实现同一个业务所有主流终端的需求，实现业务的跨终端。这样做的好处非常明显：保证了技术和业务上的一致性，提升了研发的效率和品质。

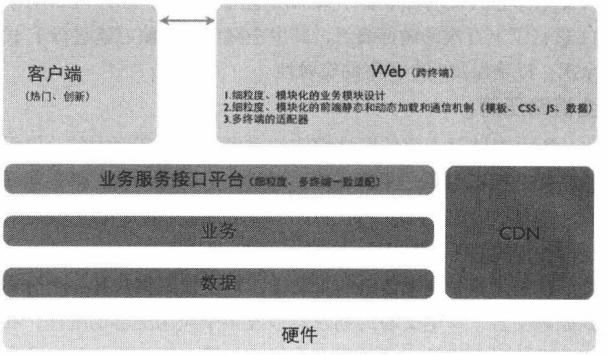

图 2 跨终端的技术架构

天猫的核心交易链路 List、Detail、Buy 都有 Mobile 版本，而每个业务的 Mobile 开发都是由对应的 Desktop 业务开发团队完成的，即 Detail 的前端团队不仅完成 Desktop Detail 的业务，还负责 Mobile 的开发维护，List、Buy 同样如此。这样做的原因是：从业务角度讲，Desktop 和 Mobile 其实都是同一产品，要实现同样的核心功能，不同终端只是因为其端展现的人机交互不同而已，同样的产品没道理由每个终端单独的团队来维护。举个例子，我们不可能因为页面需要兼容 IE 和 Firefox，就把前端团队分为一部分人做 IE，一部分人做 Firefox，对于 Desktop 和 Mobile 也是如此。另一方面，一个团队维护可以节约大量的成本，对业务有更深度的认识，只由一个团队维护开发，才更有可能从整体看待 Desktop、Mobile 开发中的问题，考虑如何优化和提升效率，也由于本身业务的一致性，开发 Desktop 的经验对于开发 Mobile 也省去大量对于业务理解的成本。基于以上原因，我们对于跨终端的实现从组织架构角度是通过一个团队来完成的，而双 11 也印证了这种做法的成功。

2013 年双 11 预热阶段有一个大型游戏“狂欢城”，往年对于这种大型游戏都是外包给外部广告公司通过 Flash 完成。但基于 2013 年跨终端的考虑，使用 Flash 技术无法支持移动端，所以我们在 2013 年采用了 Web 技术的方式实现：

■ 占比 66% 的高级浏览器使用原生 Canvas；

■ IE6/7/8 浏览器方案，使用 FlashCanvas 兼容。

关于低级浏览器兼容，我们对比过几款常见的 Canvas 兼容方案，FlashCanvas 是比较完善的方式。这对于天猫是首次尝试，业界类似的案例也非常少见。为了降低风险，我们很早就开始准备技术方案，包括通过 Web 技术把 2012 年的 Flash 版“狂欢城”完全实现了一遍。实际开发中也遇到很多挑战，例如 FlashCanvas 的一些 Bug、如何解决游戏性能效果等。从最终结果看，无论从游戏的跳失率、互动率还是转化率等业务指标都较 2012 年有了大幅提升，同时也为公司节省了一笔不小的外包费用。最重要的是，这为以后大型互动游戏的跨终端做好铺垫。最终我们沉淀了一套游戏框架，如图 3 所示。

双 11 当天，在天猫首页还有“捉猫猫”的游戏，实际上这个游戏前端只用了 1 天时间就完成了桌面、平板、手机的完全兼容，这得益于初期对于游戏的准备，也再次证明了可以通过一致的开发方案，低成本地解决跨终端问题。

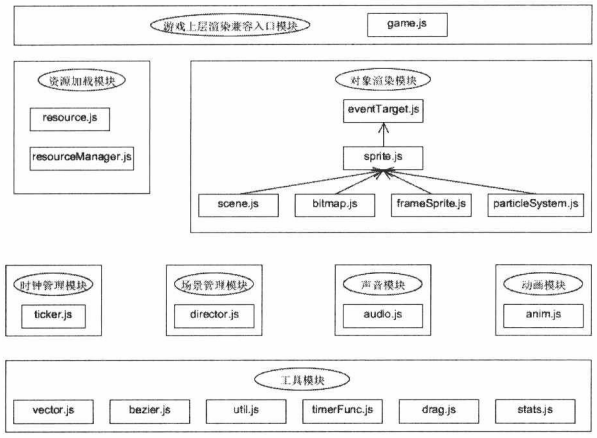

图 3　游戏框架

一致、高效、灵活、新鲜的前端架构和发布机制：MAP

跨终端的前端解决方案的前提是，相同业务在不同终端上的业务本质和人机交互本质是一致的，为了保证能把 Web 快速地推进到各个终端，需要建立统一的前端架构和发布机制，我们称之为 MAP（Tmall Front-End Architecture & Published Mechanism）。MAP 的目标不只是为双 11 服务，但每年的双 11 绝对是验证这一年前端架构和发布机制改进成果的最佳时机，满足了双 11 的苛刻环境就意味着为下一年常态化运作打下了坚实的基础。2013 年，MAP 实现了从 1.0 到 2.0 的巨大升级，目标就是成为一致、高效、灵活、新鲜的前端开发环境，如图 4 所示。基础设施的一致性是高效自动化的基础，也是前端代码能够自由流动的基础，而一致性对本质和未来的探讨是达到灵活的前提，同时保持一致性实现方案的新鲜度和先进性，不断地自我革新，是前端架构和发布机制能够长期存活的关键因素。

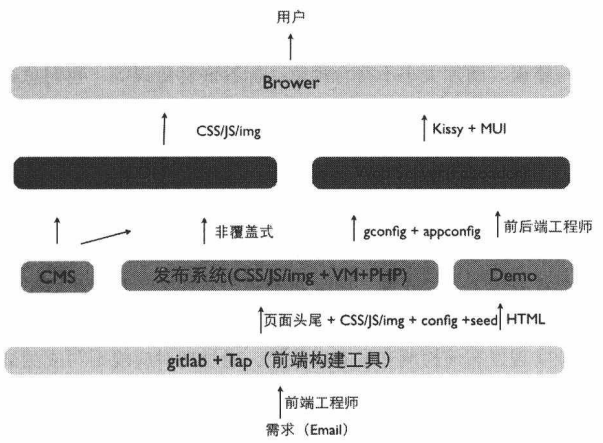

图 4　MAP2.0 的架构

MAP 2.0 的特色主要体现在以下 4 个方面。

■ CDN 静态文件全部是非覆盖式发布，且采用语义化路径。使用 CDN 存储静态文件是 Web 性能优化最关键的步骤，也导致了静态文件和动态文件发布的差异性。通过静态文件的非覆盖式发布，就能实现静态文件和动态文件发布的分离，静态文件开发完毕可提前随时发布，从而避开前后端发布无法同步而导致短时间内错乱的问题，同时，在出现问题时可以轻易地实现前端和后端各自独立回滚。我们把语义化版本号引入到文件路径作为非覆盖式发布的标识，比如查看天猫首页源文件会发现类似文件路径 http://g.tbcdn.cn/tm/fp/1.7.1/core.js，无须任何解释，前端也能很快知晓这个路径表达的含义。通过语义化版本路径能让前端和合作伙伴不借助任何说明就能知晓代码的架构和含义，这种规划不仅全面提升了认知和理解效率，也为自动化发布工具打下了坚实的基础，并与多版本迭代开发的项目管理有机地结合起来。

■ 双层扁平化前端代码组织结构和前后端一致的静态文件引用方式。整个天猫前端代码的组织基础是 Kissy（基础前端框架）和 MUI（天猫前端业务组件），通过保证 Kissy+MUI 在整个天猫前端的一致性实现，业务组件能够快速且一致地部署到全站，例如双 11 页面左侧的天猫工具栏和全站顶通。Kissy 和 MUI 都有大量组件，我们通过全站统一的配置文件实现这些组件引用的部署，这样每个页面只需要按需加载组件即可。由于性能优化的需要，首屏首次访问需在服务端直接渲染完后推送到客户端，其他内容可以是 JavaScript 动态渲染，所以天猫的前后端都部署了 Loader 机制，实现了前后端一致且灵活的加载机制（如图 5 所示）。例如，后端引用前端文件代码是 FeLoader.use(a,b)，在前端则是 Kissy.use(a,b)，其执行之后输出的代码是一致的，都类似 http://g.tbcdn.cn??1.3.0/seed.js&1.4.0/a.js&2.1.0/b.js，从这个 URL 可以看出，通过 Loader 机制能够同时在前后端自动实现一致的依赖关系处理和合并加载。

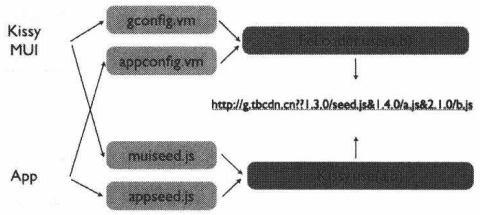

图 5　一致性的前后端按需加载机制

■ 扁平化细粒度的代码仓库和开发构建工具。整个天猫的前端代码都是通过 Git 管理起来的，每个仓库都是最小的独立业务发布单元。代码结构扁平和细粒度，保证了前端发布独立且去中心化，再结合基于 Node.js 的前端构建工具 TAP，能容易地把代码快速地发布到仓库、Demo 服务器、测试服务器和线上环境，同时能在各个发布环节进行合法性验证，保证发布质量。这一切就是因为基础机制的一致性才能够轻易地实现开发和发布的自动化，保证双 11 能够享受到快速且高品质的发布。

■ 端到端的通用接口规范。随着前端开发从兼容多浏览器到跨终端，使得开发从一套数据一个 View 层变成了一套数据多个 View 层，前后端分离变成必然，这其中最关键的点就是前后端的数据接口，因此我们建立了名为 IF 的规范。它提供了一套数据接口规范（Web 前端与后端服务的接口、Web 服务端模板与后端服务的接口、Web 前端与 Native 客户端的接口、Native 客户端与后端服务的接口与后端服务之前的接口），及一个接口工具集支持整个开发生命周期中的接口需求。通过 IF，保证了基本数据接口的一致性和合法性，从而大大避免了接口带来的风险，提升了整个前端产品的稳定性。

天猫工具栏部署在天猫全站（页面左侧），这是一个采用“框架+插件”机制开发且业务复杂的组件，它既是MAP的试金石，同时也可能是未来全站Web App的雏形。

- 由于MAP保证了全站一致的基础库，工具栏才免于对基础库做兼容处理，节约了超过10%的时间，保证了项目的高质量上线。
- 工具栏本身采用的“框架+插件”的架构思想和MAP“扁平化细粒度”的设计理念保持一致。所有插件被设计为可由独立的代码仓库管理、并可单独发布，而工具栏框架可以实现插件“热拔插”；这样就保证了部署于全站的框架的稳定性，同时又适应了工具栏业务频繁变更的需求（每个插件承载一项业务）。
- 工具栏中存在的20多个异步接口直接使用IF维护接口的定义、文档、Mock数据及校验，确保了在数十人协同开发、接口频繁改动、存在接口降级的情况下，在双11期间没有产生一起由接口直接导致的线上故障。

前端降级预案和限流机制

在双11大并发、高流量的业务场景下，前端还需要考虑如何保证用户的优雅体验及系统稳定平衡，例如后端系统当天有可能出错或响应缓慢。如何在这种情况下提供用户可用的体验，是前端面临的挑战。我们的技术方案主要从降级和限流两个维度实施。

在业务层面，我们降级掉一些不太重要、当天对用户影响小的业务，例如List的Minisite展示、List关键词的关联推荐、首页的个性化推荐等，这些在双11当天都被降级不显示，这样可以减少额外的系统压力，让后端机器优先保障核心交易链路的稳定。对于前端技术层面，在开发各个功能模块时就需要考虑模块不展示的可能，需要实现模块细粒度的“热拔插”。对于双11，需要梳理全站前端可能降级的情况，制定前端的降级的模块列表，针对这些模块进行降级和取消降级的测试。对于降级备案的执行分为两种情况：双11之前执行和双11当天判断。如前面提到的List关键词关联推荐，就属于11月初即降级；而List的小图展现的降级方案，则是在双11当天根据CDN流量情况选择执行的（最终因CDN容量较充足而没有执行）。总之，前端为双11备战的一项重要工作就是梳理全站（首页、List、Detail、登录、交易等）降级预案，这需要充分依赖于前端架构的细粒度和灵活性。

除了降级预案，还需要考虑另一种有意思的场景，就是后端接口限流。为了保证后端服务的稳定性，在后端无法支撑高流量或并发的情况下，后端会启用一种保护机制，对部分请求直接返回静态的内容，避开对后端应用造成压力。另一方面，前后端的通信接口是通过异步请求（Ajax、JSONP等）的，如果不做任何处理，限流时接口会和普通页面一样返回一个静态HTML内容，这样就会引起脚本解析报错，甚至流程无法走通。因此必须对异步请求返回特定的数据，标识这种情况，以便前端拿到标识做相应处理。对于异步请求，首先需要解决的问题是如何识别。这涉及两种方法：一种通过XMLHttpRequest发出，这种后端可以通过HTTP Header中X-Requested-With=XMLHttpRequest识别；另一种为JSONP请求。我们的做法是：对于URL中存在callback=jsonpxxx的情况即认为JSONP请求，后端识别后即可针对异步请求返回特定JSON数据，整体方案如图6所示。

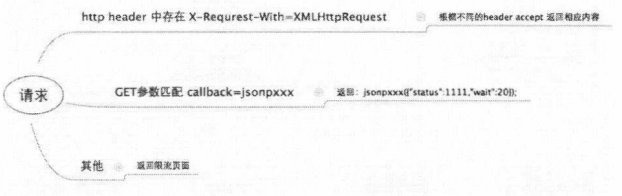

图6 限流的技术方案

举个例子：天猫首页焦点图，有一张是依赖后端接口返回数据展示的，在接口被限流的情况下，前端能根据标识使用备份的打底图片展示，而不影响用户体验；而2012年当用户被限流时只能JavaScript报错，而无法处理（如图7所示）。

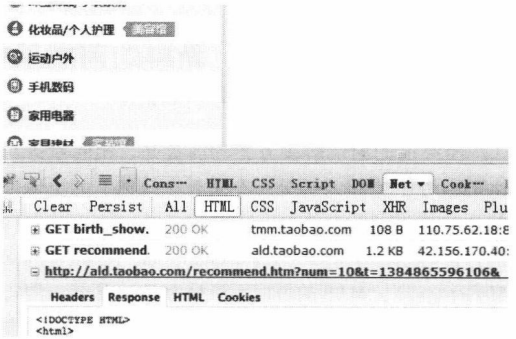

图7 不做处理时首页焦点图情况

前端质量检查和监控平台：猫须

双11活动中前端需要短时间产出大量页面，其中涉及多个角色的共同协作，面对消费者极高的要求，如何更好地保证页面质量是一个挑战。我们从几个维度来迎接这个挑战：内容填写时、发布前和发布后。对于活动页面，大量内容是运营同学在后端填写配置的，其中很难避免人为的疏忽错误，比如链接填写错误、图片大小不符合规范、标签嵌套错误等。我们总结了保证内容质量的一些规则，在CMS系统中填写完数据后，依照这些规则，对内容进行检验，来避免数据层面的一些常见错误。还有一些代码层面的问题，例如代码中是否存在安全漏洞（XSS、UTF-7漏洞、JSONP、Ajax等异步请求的漏洞），是否存在一些内部的静态资源引用等，因此在发布前还有道检查。在发布后，需要检查线上是否正常，线上系统是否有冲突。我们积累了大量这方面的规则和经验，这次将规则和经验自动化起来，借助猫须实现发布前检测和发布后监控，通过把对应规则写成测试用例，部署到在线系统，然后定时自动检查，产出功能及性能测试报告，并跟内部统一的Bug跟踪流程平台打通，提交给相应同事进行跟踪改进。实时且高效的上线前检查和上线后监控非常有效，尤其是在双11这种场景下，能快速地保证大量运营页面的品质和性能。

猫须分为两部分：操作平台和运行服务。提供操作平台的关键目的是提供各种丰富的工具自动化生成用例代码，这也是自动化思想的一个体现。例如录制交互行为、录制样式、自动创建XSS URL等，这些都可以非常方便地自动化生成用例。对

于这方面的监控需求，可以做到无需手写用例代码，这也很大程度地减少了用例代码的维护成本，大不了重新录制一遍而已。在平台上，还有一个非常重要的思想：汇聚优质的通用用例模块，增加用例代码的重复利用度，统一维护，为创建任务服务。运行服务是运行任务的服务，跟平台可以结合也可以单独部署运行，方便后续调度任务的需要。在运行时，当任务运行完成后，如果该任务下有错误信息，会及时通过邮件等方式告知到任务的创建者，推动快速解决问题。

猫须从 2013 年 9 月初开始建设，10 月初投入使用，需要在一个月左右的时间内，完成对双 11 众多会场页面、重要业务页面的监控任务。之所以能这么快地实现，是因为猫须是构建在大量业界开源项目之上的，我们依据需要进行了深度开发。

猫须采用 Node.js+Express 架构开发，采用 Node.js 的原因是其可以快速地架构和开发，非常多成熟的服务端模块可以投入使用。在数据库上采用了 MySQL，Node.js 里面有 MySQL 模块可以做很好的支持。猫须支持多种浏览器，当然也包括 PhantomJS，这个得益于 Selenium-Webdriver，但只利用它来启动和关闭浏览器，并不依赖它提供操作浏览器和页面的 API，因为猫须有自己非常丰富的 API。猫须提供的 API，如图 8 所示。

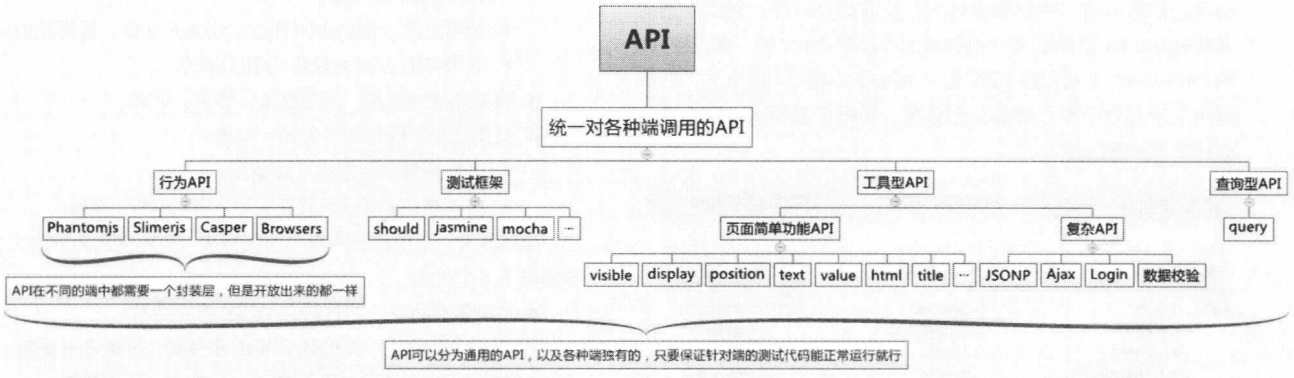

图 8　猫须的 API

总之，猫须的核心目标是上线前检测，上线后对线上页面的质量做到实时监控，及时发现问题，并及时解决问题；同时也要最大程度地降低用例代码的维护成本，创建一套适合于天猫的前端标准规则体系。

精确到秒的发布计划和实时发布能力

双 11 就是一场战争，所以在当天的安排上，所有的准备都是为这一天我们能够活着度过而做的。当天的发布有非常苛刻的时间要求（如图 9 所示），我们制定了精确到秒的发布计划，包括几分几秒需要发布哪些页面、由谁具体操作、Double Check 人员是谁、影响范围等。双 11 当天所有前端严格按照时间发布表执行，并更新每次发布状态，做到信息同步。除了组织安排上的保证，由于发布到 CDN 或服务器都存在一定延时，所以技术上还需要保证时间的准确性，整体上分成以下三种情况。

- 对于时间要求不高的发布（例如会场页面预热到正式的切换），可以通过人为、内部 CMS 系统定时发布完成。
- 对于精确的秒级业务展示，例如当天 00:00:00 秒需要出现的顶通，我们是提前发布，通过后端代码判断服务器时间展示，对于服务器每分钟都会自动校准时间，保证服务器时间的稳定性。
- 对于用户实时性要求极高的展示，例如：00:00:00 秒天猫首页出现的开幕式，单靠第二种处理还不够，因为用户有可能之前已打开首页，如果不刷新将看不到开幕式，而开幕式又持续极短，我们希望尽量让所有用户看到。对于这种情况，就需要前端 JavaScript 代码轮询判断时间来给用户展现，但 JavaScript 本身执行时间存在不准确的问题。为了保证准确性，JavaScript 不断地去服务器校准时间，以保证用户端可以准时看到效果。

战争的魅力之一就是意外，因此除了精确到秒的前端发布外，还有大量依据当天状态的临时冲锋，在满足大流量、高并发下用户继续完成购物的需求外，还需要快速地进行特别内容的开发和整站随时上线，这个过程虽然和平时几乎一样，但当时能够快速、稳定地完成就是依赖于平时所做的一致性的前端架构和发布机制 MAP，充分显示日常开发的能力。

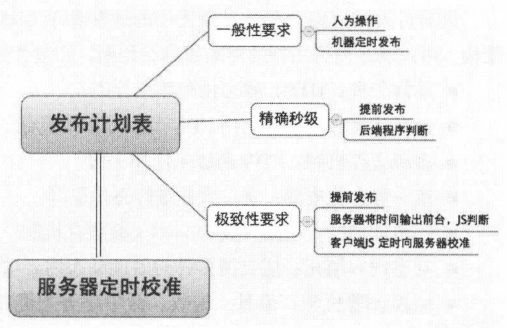

图 9　精确到秒的时间方案

总之，双 11 有类似前端降级和限流这些独特的挑战，也有类似跨终端解决方案这样的前端创新。在这个充满挑战和压力的环境中，无论是个人还是团队都得到了极大的锻炼和成长，累得要死但依旧激情满怀，这是其非常迷人的地方。

天猫浏览型应用的 CDN 静态化架构演变

文 / 徐昭

在天猫双 11 活动中，商品详情、店铺等浏览型系统，通常会承受超出日常数倍甚至数十倍的流量冲击。随着历年来双 11 流量的大幅增加，每年这些浏览型系统都要面临容量评估、硬件扩容、性能优化等各类技术挑战。因此，架构方面的重点在于，如何能够利用合理成本应对瞬间飙高的峰值请求，并确保活动完整周期中系统容量的可伸缩性、用户响应时间的稳定性，以及外部依赖系统出现问题时的高可用性。此外，作为最主要的页面流量承载体系，架构方面还需考虑防爬攻击、流控容灾等安全、稳定的需求，并综合衡量网络带宽、硬件成本、缓存效率等各方面要素，找准平衡点，从而达到以不变应万变的理想效果。

演进

为此，自 2011 年起，以天猫商品详情系统为代表，天猫浏览型系统在架构上的主要工作之一就是通过静态化技术实现了动静态信息分离、利用缓存技术存放静态化内容、利用少量动态数据异步加载填充。整个过程历经单机静态化、统一缓存接入，到 2013 年双 11 前彻底 CDN 化三个阶段（如图 1 所示），有效解决了缓存命中率、流量自然分布、系统扩容简化、用户端响应速度等关键问题。

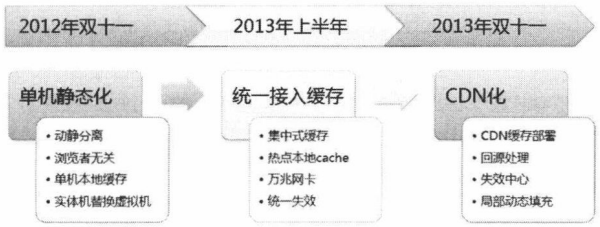

图 1　CDN 化的三个阶段

目前，天猫浏览型系统最新使用的这套基于 CDN 的静态化架构，可以满足高可用持续伸缩的原始预期，并包含如下特性。

- 动静分离：HTML 静态化和热点分离。
- 分布式缓存体系：利用 CDN 节点分布式缓存。
- 多级缓存机制：CDN 两级 + 应用一级。
- 统一静态化集群：统一服务端静态化集群。
- 一致性维持：主动失效 & 自动失效缓存机制。
- 动态内容填充：能支持多种时效性动态内容填充方式。
- 监控预警机制：流量、失效、命中率等关键参数实时监控报警。

本文将针对这一优化历程，就主要技术挑战、架构改造策略、最终优化成果做一个总览式的介绍，并重点对 CDN 化过程中整体架构的演进、缓存失效机制、动态内容填充等具体要点进行论述。

第一阶段：系统静态化

早期天猫浏览型系统大多采用简单架构，实现一层很薄的前台应用。以天猫商品详情系统为例，针对商品、用户等访问量较大的数据中心接口模式改造为应用 Client 端缓存前置，同时普遍使用页面高速缓存（PageCache）来降低后端系统压力，使得整体可支持应用水平扩展不受限制。这一阶段系统面临的主要问题和挑战包括以下几点。

- 应用服务器瓶颈，页面渲染带来的 CPU 开销巨大。
- 单纯基于 Java 端的缓存已基本覆盖，整体性能提升空间有限。
- 水平扩容只能支持容量线性提升，难以满足大促时期井喷式流量增长，扩容成本高。

从问题看，基于原有动态浏览型系统模式而优化的瓶颈很难规避，例如以下几点。

- Java 应用服务器端必要开销，包括：
  - 涉及页面内容的字符串查找、替换、拼接等；
  - 元数据获取的网络开销；
  - Servlet 本身的性能瓶颈。
- Web 服务器端，包括：
  - 模块过滤，例如访问日志、Cookie 打点、繁简转换；
  - 大 HTML 页面本身的 GZIP 压缩等。
- 突发流量的抵御，例如攻击、秒杀、大促。
- 已用优化手段达到了边界，包括：
  - 可使用缓存的地方已经使用；
  - 服务端 CPU 能力已优化完毕（模板解析、压缩）。

总体来看，必须从架构着手彻底解决。架构优化的方向上，考虑以下 3 个方面。

- 改变缓存方式，直接缓存 HTTP 响应结果。
- 改变缓存位置，直接基于 Web 服务器，屏蔽业务逻辑。
- 基本原则，缓存空间足够大、无单点、易于维护。

为此，2012 年起正式启动了动态浏览型系统的改造项目，通过静态化手段解决上述问题。即基于业务把原动态系统中的内容做动静分离，对浏览者无关部分做缓存，动态内容做 CSI 填充。具体考虑从三方面重点着手展开：动静信息分离、静态化缓存方式，以及缓存失效机制。图 2 为一期静态化整体架构。

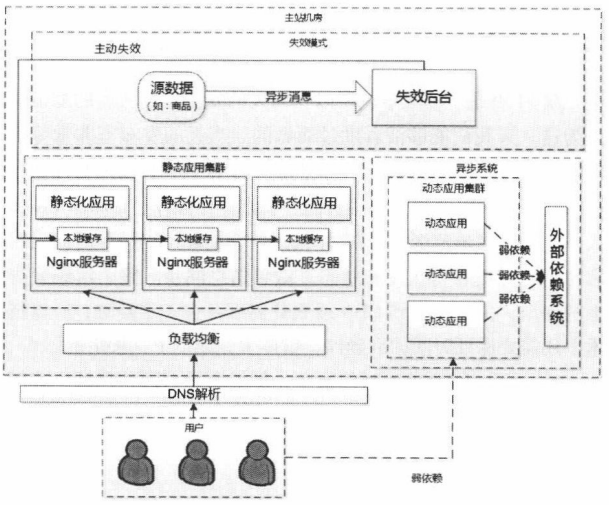

图 2　一期静态化整体架构