

[架构师]

# 双11特刊

# 电商大促技术探秘



# 卷首语

作者 蔡芳芳

今年蚂蚁金服的双十一技术媒体沟通会上，蚂蚁金服副总裁胡喜说“天猫双11”已经越来越常态化，而这背后其实是技术在变得越来越智能化。

如果将2016年称为人工智能元年，2017年则是人工智能的应用之年，巨头如阿里、腾讯、京东都早早投入AI浪潮之中。今年双11，蚂蚁金服将人工智能应用于容量预测、弹性计算、智能客服、故障监控等方方面面；京东的全链路故障演练、智能补货、销量预测、供应链体系优化也都是通过人工智能技术来实现的。AI已经成为双十一电商大考的新利器，InfoQ希望通过报道这些公司的实践案例，给读者呈现大公司领先的创新技术和实践经验。

但对新技术的探索和寻求改变从来不只是局限于巨头。苏宁自研的穆加决策分析平台引入了机器学习，具备问题自动判断和定位的能力，实现了“监”与“控”的结合；国美借助实时系统、深度学习及在线学习等技术打

造个性化精准推荐，大幅提升了CTR与CVR转化率；蘑菇街将图像搜索技术和图像标签技术应用于大促场景，从而提高了商品管理的效率、改善了用户体验；除此之外，蘑菇街今年还站在了小程序的风口之上，本次双11蘑菇街小程序的成交数据远超APP，在其他电商陷入流量增长停滞转而抢夺线下时，蘑菇街找到了一条不一样的路。

可能会有读者觉得这些公司与阿里、京东比起来流量差了不少，但从另一个角度来看，中小型公司遇到的技术问题可能与大公司有所不同；再者，大公司的规模和流量也非一蹴而就，在同样体量、不同时机之下，是否能从不同角度寻找问题的解决方案？

对InfoQ来说，报道的焦点不局限于电商巨头，只要具有技术创新点和可以学习借鉴的实践经验，就值得报道。我们希望通过挖掘和分享不同公司的技术亮点，推动整个电商生态、甚至不同行业之间的技术交流。

就像前几天刘强东在周鸿祎的新书发布会上说的那样：“以前的互联网是野蛮生长的，很难看到清晰的规则，这两年情况有所好转，但两极化越来越严重，流量越来越集中。今天全球的互联网行业都在走向垄断，这是危险的事，所以不仅企业要战斗下去，还要不断呼吁，推动整个行业走向文明。这样才能给无数新创业者留下机会。如果十年二十年后，中国还是BAT，还是京东、360这些公司，对这个国家绝对是个不幸的事情。”

InfoQ希望能通过传播技术内容推动多方竞争、加速技术创新。今年是InfoQ双十一专题内容策划的第四年，读屏时代带来了很多新的挑战，如何将好内容以更好的形式呈现给更多的读者，是我们近来一直在思考的问题。今年我们尝试将技术内容转换为语音并推出了极客时间APP，思考和创新不会止步于此，希望来年双十一专题能以更棒的内容形式与大家再见面。

# 助力人工智能落地

2018.01.13 – 01.14 · 北京国际会议中心

人工智能已不再停留在大家的想象之中，各路大牛也都纷纷抓住这波风口，投入AI创业大潮。那么，2017年，到底都有哪些AI落地案例呢？机器学习、深度学习、NLP、图像识别等技术又该如何用来解决业务问题？

由InfoQ举办的AiCon全球人工智能技术大会上，一些大牛将首次分享AI在金融、教育、电商、外卖、搜索推荐、人脸识别、自动驾驶、语音交互等领域的最新落地案例，以及在落地过程中的那些痛点和难点，一些技术细节该如何操作，有哪些避坑经验，应该能学到不少东西。

## 部分演讲嘉宾



颜水成  
360人工智能研究院  
院长及首席科学家



山世光  
中科院智能信息处理  
重点实验室常务副主任  
中科视拓董事长/CTO



袁进辉（老师木）  
一流科技  
创始人



刘海峰  
京东商城  
总架构师&技术VP



洪亮劫  
Etsy  
数据科学主管



于磊  
携程  
基础大数据产品团队总监



张浩  
饿了么  
技术副总裁



尹大朏  
摩拜单车  
首席科学家

## 精彩案例抢先看

摩拜 | 如何使用人工智能实现单车精细化运营

知乎 | 如何使用机器学习实现News Feed正向  
交互率提升100%

国美 | 推荐引擎与算法持续部署实践

微博 | 深度学习在红豆Live直播推荐系统中的应用

Tutorabc | 大数据和AI之路

饿了么 | 机器学习和运筹优化在外卖行业的应用实践

第四范式 | 如何利用大规模机器学习技术解决  
问题并创造价值

微信小程序 | 商业智能技术应用实践

爱奇艺 | 自然语言处理和视频大数据分析应用

8折 限时优惠进行中，每张立减720元  
截至2017年12月15日前 团购享受更优惠

邮箱：hedy.hu@geekbang.org 电话：18510377288（同微信）



购票请联系



扫码关注大会官网  
获取更多大会信息



# 极客时间

重拾极客精神·提升技术认知

## 「专栏订阅」

每天 10 分钟，邀请顶级技术专家，探究技术本质，解读科技动态。

## 「极客新闻」

每天早上 8 点，朝闻技术天下事。

## 「热点专题」

最前沿的专题，最独特的视角，最风趣的解读。

## 「二叉树视频」

一档属于技术人的直播和短视频节目，记录与时代并行的技术人。



关注我，获取更多干货



# 目录

- 07 天猫 11·11：蚂蚁金服如何用小团队支撑数亿人买买买？**
- 18 国美 11·11：大促场景下的国美智能推荐系统演进之路**
- 28 蘑菇街 11·11：图像算法在电商大促中的应用浅析**
- 39 苏宁 11·11：从 0 到 1，苏宁 API 网关的演进之路**
- 59 当当 11·11：高可用移动入口与搜索新架构实践**
- 75 有赞 11·11：全链路压测方案设计与实施详解**

# 天猫 11·11：蚂蚁金服如何用小团队支撑数亿人买买买？

作者 蔡芳芳



又到了一年一度的“天猫双11”，外行看热闹（剁手），内行看门道（技术）。每年“天猫双11”都会创造不少与“交易量”相关的世界纪录，而这些世界纪录的背后则有改变世界的技术作支撑。作为买买买的重要一环，支付宝是如何撑起12万笔每秒的交易量的？InfoQ小编带你走进蚂蚁金服一探究竟。

下面这张图，记录了支付宝历年双11的峰值交易数据。

从2010年的每秒300笔，到2016年的每秒12万笔，交易笔数提升的背后，是蚂蚁金服技术能力被“逼”着快速升级。



## 从人肉云计算到智能化云平台

### 曾经支付宝距离数据库崩溃仅剩 4 秒

支付宝刚起步时，技术远没有今天那么受重视，原因很简单，不需要、没必要。在创业之初，2004 年时，支付宝还是淘宝中的一个结算部门，淘宝的会计人员用两台电脑和一张 Excel 表就能进行结算。那时每天的交易金额是三位数，全天交易笔数只有十几笔，如果分摊到每秒钟，则约等于零。

2009 年是淘宝第一次搞双 11 大促，那时双 11 还不像现在这样广为人知，所以 2009 年的交易量并不大。在此之前，支付宝也刚完成二代架构的升级改造。在二代架构做完之后，支付宝的技术团队感觉能解决的技术问题都已经解决了，很多人认为未来系统也许就可以这样发展下去。因此在 2010 年双 11 大促之前，支付宝的系统规划是按照每年增长 100% 余量预估的。

但谁也没有预料到人们买买买的疯狂程度，二代架构上线并没有多久就迎来了严峻的考验。双 11 当天零点刚过，支付宝的业务量快速攀升，每秒交易量直接飙到平时峰值的三倍——每秒 500 笔，而且居高不下。这时大家才意识到，系统容量并不足以支撑当天的交易量。

情急之下，支付宝技术团队开始不停地“搬资源”、“砍业务”，东拼西揍地保障系统不崩溃，这一窘迫的过程事后又被大家戏称为“人肉云计算”。

算”。直到当天的 23 时 59 分 30 秒，眼看 2010 年双 11 大促就要结束了，突然核心账务系统报警。千钧一发之际，技术团队紧急将内部对账用的会计系统应用杀掉（可以在双 11 过去之后再用交易数据恢复），将资源释放出来，这次双 11 才算有惊无险地度过，而这时距离数据库崩溃仅剩 4 秒。

## “双 11”逼出来支付宝的三代架构

从 2005 年开始，支付宝的技术架构经历了“烟囱型”、“面向服务型”、“云平台型”三个时期。支付宝的三代架构可以说是和双 11 彼此成就的。



支付宝的第一代架构就像一个个独立的“烟囱”，没有基础架构可言，做一个业务就竖起一个“烟囱”。“烟囱”之间的关联性不大，每做一个新业务就要对一个烟囱进行手动改造，而支持主要业务的“大烟囱”则经历了无数次改动。

第一代“烟囱型”架构能够满足小团队开发、业务快速试错的需求，但是这个架构无法支持大团队的分布式开发。与此同时，它的部署也是集中的，核心系统就是一个集群，数据库也只有一个。随着业务量的上升，这样的数据库和集群很快就会达到极限。为了满足业务扩展的需要，分布式架构改造被提上了日程。

当时，分布式系统在互联网的应用并不罕见，规模较大的互联网公司系统都分布化了。但是由于金融系统对于稳定性和安全性要求较高，分布

化尚无先例。对于金融系统来说，任何改进都要先保证用户账目上的钱分毫不差，在系统数据分离之后，保证系统之间业务处理的一致性就成了核心问题。

2007年初，分布式系统技术逐渐成熟，特别是大规模SOA系统中的分布式事务处理标准逐渐明晰，支付宝随之启动了分布式改造工作。从2007年开始，支付宝陆续花费了三年左右对整个系统进行分布化，将原来的大系统拆成了一个个分布式的“服务”。由此，支付宝夯实了分布式事务的基础设施，所有的业务都可以分开来做，系统具备了可伸缩性，如果遇到业务峰值，就可以增加资源。当然，这时增加资源还需要人工搬运。但支付宝的第二代架构已经为第三代架构“云支付”打下了基础，因为如果使用云计算，系统首先要分布，只有这样才能用很多小型机器、云资源作为支撑。

二代架构改造完成之后，淘宝开始了每年一度的双11大促。对于支付宝技术来说，2010年是一个拐点，这一年，交易数据的峰值比此前翻了三倍，也正是这一年惊险的双11让技术团队意识到，业界现有技术和传统架构已经无法满足支付宝的发展需求。

支付宝技术团队尝试了一种新的对策——分布式“异地多活”架构。蚂蚁金服副总裁胡喜将其比喻为“拆掉了高端中央收银台，换成了分散在商场各个角落的无数小型计算器”，每台计算器虽然不如单一中央收银台厉害，但个个都能记点帐，同时支付宝为分散在各处的计算器设计了相互关联的逻辑关系，使它们互为补充、互相备份，从全局上保证运算可靠，因而任何单个计算器的故障，都不会影响整个系统的正常运行。这是这种架构中最核心的云计算能力。

从二代架构过渡到三代架构，支付宝又花了三年时间。在此期间，支付宝开始自主研发中间件、数据库、大数据平台。第三代云支付架构完成了两方面的改造：一方面是在底层使用云计算技术；另一方面是在上层把服务变成云服务，如此一来，建立在这个架构之上的业务增长就不会受到限制。

正是这个云计算架构，支撑了近几年淘宝大促的交易，也使后来的“天猫双11”能够平稳地进行。

2016年双11蚂蚁金服在技术保障上有两个亮点，一是整个支付核心链路，包括交易、支付、会员、账务都运行在自研数据库OceanBase上；另一个是蚂蚁自主研发的弹性架构，能够利用全国多个城市的云计算资源，从而支撑12万笔/秒的支付峰值（2015年的1.4倍）。

弹性架构具备在云计算平台上快速伸缩容量的能力，50%流量基于云计算资源弹性伸缩，能够快速扩充支付容量，从而优化运维成本。理论上可以做到每秒百万级的交易支付能力。

## 2017年支付技术保障新亮点

### 1. 离在线混合部署

今年蚂蚁金服推出了离在线混合部署，使用的服务器资源有25%是自有的，55%在云上，20%是离线资源。今年天猫双11交易高峰期，一部分交易将首次跑在临时“征用”来的存放和处理分析离线数据的服务器上。



说是临时“征用”，但绝不仅仅是“征来就用”那么简单。因为处理离线数据（即处理每天数据分析报告以及当日账表）和处理在线交易（即处理支付宝用户在线的海量实时交易）是完全不同的任务，这也决定了处理离

线数据和处理在线交易的服务器的配置相差悬殊。

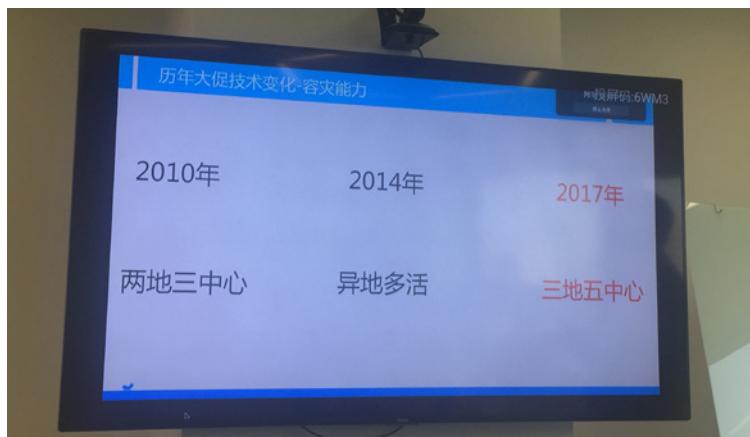
如何在交易高峰期，以秒级的速度将在线服务器的各种软件、应用转移到离线服务器中？这背后起到关键作用的，是容器技术和统一资源调度的能力。容器技术相当于 标准化改造，有了这个标准化改造的能力，各种配置不一、可以暂时放下手中的活伸出援手的服务器，都可以在各种大促场景派上用场，帮助疏通与分流。据估算，离在线混部技术能为 2017 年天猫双 11 减少近两千台服务器的投入。

## 2. 超级会计师 OceanBase

2016 年天猫双 11，12 万笔每秒交易峰值的背后，是“超级会计师”OceanBase 在发挥关键作用。

OceanBase 是一个海量数据库，可以存放千亿条以上的记录，单台普通的服务器每秒可以处理百万笔事务，平均一次花的时间仅在毫秒级别。

但意外的发生：服务器机房所在城市断电、连接机房的光纤被挖断、再好的机器也有出故障的时候……所以，金融机构普遍采用两地三中心来部署机房。



为了达到更高的可靠性，OceanBase 今年开始把数据同时放到三个城市，做五份数据备份，进一步降低了出错的可能性。

三地五中心，不仅仅是多了一个城市两个机房，蚂蚁金服还尝试使用一个新的选举协议：在出意外情况时，谁得到大多数投票，谁就当选主库。主库不再是固定的，每个被选举出的主库都是临时的，且它做的决定

必须得到半数以上的同意才能实施。举个例子，一个主库刚被选举出来，它所在城市的光纤突然被挖断了，它会立即被票出去，做不成老大了，它做的决定也会全部被宣布无效；同时，一个新的主库会被票选出来，保证线上交易的顺利进行。

采用三地五中心的方案后，主库突发故障或者任何一个甚至两个机房同时断电、断网，业务都能在极短时间内自动恢复，不需要任何形式的人工对账。

### 3. 金融级图数据库 GeaBase

除了 OceanBase，蚂蚁技术团队还自主研发了一个叫 GeaBase（以下简称 GB）的金融级图数据库，它通过特有的数据组织方式和分布式并行计算算法，可以在几十毫秒内彻查目标对象的多跳资金转移关系、设备关联关系等组成的复杂网络，从而迅速锁定目标关联、识别欺诈。

GeaBase 是蚂蚁金服风控系统背后的关键技术支撑之一。刷单党、花呗套现党、羊毛党、欺诈等行为，都依靠 GeaBase 来识别。胡喜介绍道：“风控从最早偏向于规则架构，到后来规则加模型架构，现在向 AI 转，这也是双 11 相关的关键能力，背后承载的是我们在分布式金融交易之外的金融级实时决策能力。”

## “未问先答”的新客服，开启主动服务模式

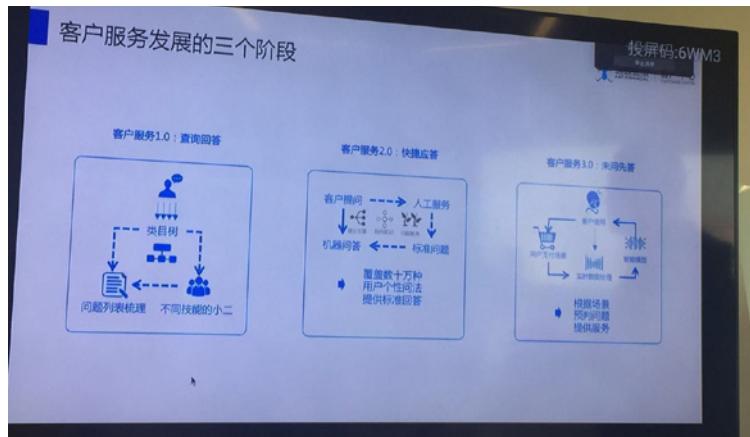
双 11 大促当天，随着交易量暴增，使用中遇到问题的用户数量也会大幅增加。不知道在你的想象中，支付宝会有多少客服小二？

记者走访了一圈蚂蚁金服 Z 空间大楼，这里的淘宝小二只有 600 人左右，算上成都团队和外包团队，也不过小数千人。而且，蚂蚁金服智能客服负责人子孟说道：“淘宝天猫平台业务量逐年增长，但是我们客服人数没有增加，反而还减少了。”

而这背后，是蚂蚁金服客服技术的进步史。

子孟介绍道，客户服务分为以下三个阶段。

### 客服 1.0：查询问答



在这个阶段，更多是查询类的事情，把很多回答的内容做成一个类目树让大家查询，不管在热线电话还是在 PC 端、APP 端，多数情况下需要大家逐层挖掘。比如我们拨打客服电话经常遇到的“xx 请按 1， xx 请按 2……人工服务请按 0”，要找到精确的问题分类或人工客服，不得不先听一段无比冗长的录音，导致查询效率底下，难以找到准确的答案。

## 客服 2.0：快捷应答

这一阶段是互联网企业通常会采用的手段，包括蚂蚁金服在很多场景下也会切换到这个手段，就是快捷应答。它包含两个点：一是更多在 APP 端、PC 端用问答机器人解决用户问题，通过用户问题可以快速识别问题分类，并指向某一个具体答案或者人工服务；二是在传统的电话过程当中，减少按键输入，更多使用语音交互方式。

## 客服 3.0：未问先答

很多时候客服的角色是相对滞后的，要等用户找上门来提出问题，甚至反复不断提出要求才能够回应，我们认为极致服务应该把事情做到事前，在用户可能遇到问题的时候提前化解他的障碍和疑问。

前一阶段积累了很多数据和用户行为的数据，我们现在希望推出的叫做“未问先答”服务。

在所有服务渠道中，我们不断依据用户的实时行为数据，经过学习和分析，在用户没有开口的时候就知道他可能想问什么问题，更快速地解决

问题，这是“未问先答”这样一个技术在整个服务中的角色。

“未问先答”服务使用效果如下（整理自真实录音）：

智能客服：欢迎致电蚂蚁金服，为给您提供自助或人工服务，请简单描述一下您的问题，请讲。

用户：我想问一下那个余额宝里面的那个一万块的话一天是多少利息？

机器客服：您是要咨询余额宝的收益如何计算，对吗？

用户：对！

机器客服：您的解决方案已经发送，你可以登录手机支付宝在首页进行查询，请问还有其他问题吗，

没有问题请挂机，如需人工服务请按 1。

“未问先答”还有其他玩法，比如用户在支付宝里反复操作、研究一个功能，但是多次操作后还没有成功，那么智能客服就会自动弹出，询问是否遇到困难，然后给出方案帮助客户解决。

## “未问先答”的技术原理

“未问先答”如神算子一般的预测能力背后，离不开数据采集、算法加工和持续反馈学习。



大量 用户行为数据 是一切的基础。蚂蚁金服为实现“未问先答”而采集的数据主要有以下几种：一是通过客服加工过的有规则的精准因子，即这个用户具备什么样的特征，意味着什么样的问题；二是大量用户在支付宝操作、点击、页面跳转行为轨迹的特征；三是客户在服务渠道上咨询过什么问题、求助过几次；四是用户在支付宝、服务端描述过的，和他的需求相关的文本信息。

通过深度神经网络 算法加工 数据。不同数据加工方式不一样，比如轨迹类数据使用 RNN 模型，从而能更好地表现建模时间的先后顺序；对于用户画像和人工设计的精准因子会使用全连接的神经网络；对于文本数据则使用注意力模型。本质上就是在海量特征和标类问题之间进行匹配，精度相对比较高。

持续不断的 反馈学习。一次性做完可能很容易，但是结果通常不会一次就达到理想水平，因此需要利用反馈数据进行自学习优化。正负反馈都能进一步优化模型的训练，整个过程是数据闭环和自学习的过程，蚂蚁金服三月份就上线了这一模型，但是运转了约半年以后这个模型才相对成熟，甚至很多数据采集其实很多年前就开始了，但也经过了 1-2 年的沉淀才真正将数据投入应用。

今年，智能客服的“未问先答”技术首次运用于双 11。在人工智能的帮助下，新客服系统能预判用户可能碰到的问题，从被动型服务（等待用户来提问）转为主动型服务（提前预知用户可能存在的问题并提供解决方案），进而提高服务效率和用户体验。

## AI Everywhere

沟通会上，胡喜回忆起三四年前的双 11，感慨万千。他说道：“以前的双 11，技术保障团队差不多三四百人，从年初开始准备，这些人很多事情都不做，就是要支持双 11，一次双 11 做完以后非常累。”而今年，智能化系统已经可以辅助人工进行系统容量预估和自动扩容。

除了容量预测和智能客服，人工智能也被应用到了双 11 的故障监控中。所有系统运行数据输入相应的机器，由机器进行训练，训练后生成模型，系统运行时模型输出的值是否合理不再由人来判断，而是由机器自动判断，从而快速发现系统异常。

胡喜表示，今年是 AI 的实验期，对于金融系统来说最大的挑战是可靠性和安全性，如何在保证系统智能化的同时提升系统的可靠性和安全性，一直是蚂蚁金服追求的方向。“今年蚂蚁金服已经把 AI 能力先应用在



一些点上，比如故障预测、容量预测等，明年我们还会推出一个内部叫做免疫系统的平台，专门做故障自动发现和恢复的事情。”

胡喜感叹：“2017 年 ‘大考’，All in 支付技术保障只需要一个弹性化的小团队，而无需占用大量人力。这意味着‘天猫双 11’越来越常态化。翻看几年前双 11 的老照片，照片里满公司的帐篷、睡袋，感觉已经是很久远的事情了。”

这背后，是技术在变得越来越智能化。

# 国美 11·11：大促场景下的国美智能推荐系统演进之路

作者 杨骥



国美早期的推荐产品，90%以上的场景是靠平台运营人员和工程师依靠业务知识进行手工配置，策略投放也是基于场景相关性的固定槽位展示，千人一面。近几年，伴随着业务的发展，尤其是实现线上线下打通后，国美互联网基于双线平台、商品和服务，利用互联网技术，以“社交+商务+利益共享”的共享零售战略向用户赋能，在后电商时代走出共享零售的新路径。同时，作为与用户交互的排头兵，国美的推荐系统也嵌入到了商品、美媒、美店等核心场景，将实时个性化推荐的购物体验带给用户。而业务场景的迅速展开和大数据的积累也促使推荐架构和机器学习算

法进行了持续地升级和迭代。

本文详细地介绍了 2016 年以来，由 11.11 大促驱动的国美个性化推荐系统中核心技术的演进历程。

## 一、实时系统铸成个性化推荐基石

2016 年 11.11 大促，我们当时的实时计算平台采用比较主流的 Kafka+Storm、辅以 Spark Streaming 的架构。在 11.11 当天，用户实时行为分析、商品 & 店铺多维度信息更新等功能支撑了实时推荐。虽然实现了推荐结果的实时性，但是从 11.11 使用过程中来看，这种架构还是有较多的不足，比如：

1. Kafka、Storm、Spark Streaming 支持“实时 / 近实时”信息收集和计算（此外推荐系统还需要 Hive/Hadoop 平台进行一些离线数据和模型的生产），但是整个数据生产的链路太长，组件太多，在稳定性、灵活性和扩展性方面问题很多。大促过程中出现了几次因为计算任务堆积而造成系统资源吃紧，不得不临时降级的情况。
2. Spark Streaming 和 Storm 由于各自的特点，不适合做机器学习模型所需要的低延时实时特征计算。
3. 系统组件太多，维护成本很高。

因此 2016 年 11.11 大促之后，我们就根据实时计算平台暴露出的问题，规划了下一步的研发方向，即实现统一的实时 & 离线计算引擎，将数据生产、特征计算和模型训练放到一起来做。通过调研，相比于 Spark、Storm 等平台单一的数据处理方式，Apache Flink 同时支持数据批处理和流式处理。因此采用 Flink，就能够同时支持在线和离线数据生产，大大简化了多系统部署和运维的成本。

今年 4 月份，我们确定了 Apache Flink 作为实时计算引擎的技术选型，并且在 Flink 上进行了二次开发，以满足国美推荐和搜索的业务需求。

经过半年多的开发和测试，新的实时计算引擎于今年 8 月底上线，实现了如下几个重要功能：

1. 用户行为亚秒级反馈：这个功能在推荐系统中的重要性不言而喻，包括商品 / 店铺召回、特征计算、展示过滤、用户画像更新等，都需用户行为数据作为支撑。
2. 实时特征计算：无论是候选集训练，还是线上排序（Offline/Online Ranking Model），要想提高模型精准度，必需引入实时特征。如果采用传统的方式进行处理，不但需要在后端服务引擎中多次调用各种数据接口，从而增加了系统整体的 I/O 开销；同时特征计算还很耗费推荐引擎的响应时间，造成线上服务延时较大。因此我们开发了一套特征计算框架，将底层特征（即初级特征）生产这种计算相对密集的任务放到 Flink 平台上进行，不但降低了算法工程师的开发难度，同时也有效降低了推荐引擎的负载。
3. 算法效果实时评测：通常 A/B 测试都是以天为粒度进行，效果反馈不及时，影响策略的及时调整。使用实时计算引擎，我们能够在 11.11 大促期间对策略投放的效果进行小时级的跟踪分析，显著地提升了流量的使用效率。

通过国美 2017 年 9 月和 10 月期间的几次促销活动实践反馈，新的基于 Flink 开发的实时计算引擎达到了预期的设计目标，并且会在今年 11.11 大促期间发挥出重要的基础支撑作用。

## 二、特征系统重塑引擎核心

排序是个性化推荐展示非常重要的一环。尤其在国美互联网三百多个推荐场景中，如何将候选集中的商品放入合理的槽位，将直接影响用户的购物体验。国美在排序方面也经历了从人工运营到规则排序，再到机器学习算法排序的演进。在实践过程中，我们发现 特征生产、存储和调用 三个环节直接影响算法排序的最终效果。

2016年上半年，我们在完成推荐引擎架构升级的同时，也进行了规则排序的大规模迭代和优化，取得了非常明显的效果，关键展位（比如首页猜你喜欢、详情页搭配购、加购成功页等）整体CTR提升31.6%，CVR提升11%。在此基础上，我们加紧研发三个月，并在2016年11.11大促期间上线了基于机器学习的CTR预估排序模型，大促期间关键展位整体CTR同比提升幅度91.2%，CVR转化率提升34.3%，效果提升可以说非常显著。

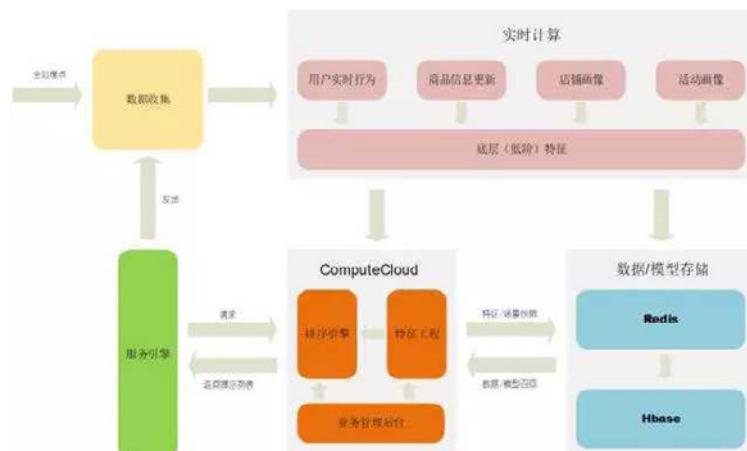


图 1. 国美个性化推荐数据流转简图

但是在整个大促期间，我们也发现特征生产方面的问题给系统带来的瓶颈：彼时特征生产的任务都压在引擎本身，也就是说，推荐服务首先调用用户行为，然后根据用户行为取商品、店铺、活动的相关信息，最后根据线上部署的机器学习模型（此时还没上线Online Learning排序模型），将这些信息转换成需要的特征，这个过程中要反复调用多个数据接口。如果模型维度较大，则进一步加大引擎本身的负载，所以流量很大时，算法排序的性能就有些吃不消了，为了保证推荐服务的可用，必须进行相应级别的降级处理。另外，响应用户请求之后，特征就相当于被“抛弃”了，接下来进行模型训练和更新的时候，还需要从仓库中重新抽取数据，重新“恢复”一遍特征，浪费了宝贵的计算资源，也影响了算法工程师的迭代速度。

2016年11.11大促之后，我们立即着手进行特征计算和存储系统的搭

建，并将这个项目和实时平台的开发同步进行。

我们对特征系统的性能要求就是高并发、低延迟、高可用，同时兼顾实时、近实时和离线特征的计算和存取。经过了一段时间的探索性尝试，我们最终确定了自研计算中间件 +KV 存储的方案。特征系统和实时计算引擎共同组成了特征生产平台。整个平台的数据计算流程为：

1. 其中以上文提到的以 Flink 为主的实时计算平台收集各种消息并进行初步处理，同时计算出一些底层特征（主要是一些统计性和描述性的特征，不产出二阶以上的特征），然后将这些实时信息和特征“Push”给计算中间件——“ComputeCloud”。
2. ComputeCloud 会和模型 & 特征管理后台时刻同步，然后根据收到的实时信息和底层特征计算出更高阶的特征。比如我们某个场景使用的排序模型，会利用 GBDT 的特征产出作为中间特征，那么计算模块会将相应的底层特征（包括从实时引擎推过来的实时底层特征，缓存在计算中间件中的近实时和离线特征等）输入 GBDT 模型，然后将各子树的叶子节点输出作为特征向量（相当于做了特征嵌入处理），提供给最终的排序模型使用。
3. KV 存储主要使用 Redis+Hbase 集群，作用是保存 ComputeCloud 中生产的特征，有两个主要目的：对 ComputeCloud 中的各种数据和特征进行备份；进行“特征快照”，将特征的每一次更新都打上时间戳，然后批量导入数据仓库，这样接下来进行模型更新或者新模型训练的时候就不用再次进行数据和场景恢复了。

目前特征计算和存储系统已经上线，并且在国美最近的几次大促中经历了考验，从效果上看，上线之后个性化推荐服务的平均响应延时降低了 40% 以上。

### 三、机器学习模型稳步升级

上面两项内容都是个性化推荐系统的基础设施，有了稳定可靠的系统，那么在模型层面上，国美如何应对 11.11 大促场景的考验？众所周

知，推荐系统在模型方面主要分为两个比较重要的范畴，即“召回”和“排序”。

### 3.1 多样化的召回模型

因为电商平台有着海量商品，在千万甚至亿级的商品池中如何“选品”，是非常关键的一个环节。国美推荐最早的召回模型大部分都是基于规则，比如品牌、品类等维度的热销、新品排行等，但是这些候选集数量较少、类型单一。为了提供更为丰富的召回结果，我们进行了大批量快速的迭代试验，最终确定了“item2item”、“搭配购”、“Low-rank Model”三大类模型。

其中“Low-rank Matrix”模型（比如 SVD、SVD++、RBM 等），我们在实践中也走了一些弯路：真实场景中的商品推荐，不可能只用一个模型就能搞定，需要针对不同的商品或者不同的人群进行训练。比如我们会针对“3C 大类”或者“日用百货”分别训练模型，因为不同大类的用户行为样本数量在量级上有差异，同时用户在不同大类的“协同”行为可比性不强。此外在 召回时还需要考虑用户的历史信息（包括用户画像），从而减少召回的计算量，避免巨量的笛卡尔积运算造成模型不可用。

2016 年 11.11 大促有近 50 个召回模型投入使用，有效地提升了推荐效果的多样性。目前，我们又在原先模型的基础上加入上下文特征，即根据用户和商品构建特征，然后进行训练。根据近一年的迭代优化与线上测试，召回模型池的整体效果又有提升，同时我们对一些效果一般的模型进行了下线处理。目前，国美推荐系统中常用的召回模型大致有 30 个左右，今年 11.11 大促会是这些模型一个非常好的“演兵场”。

### 3.2 E&E 机制加快选品速度

上文提到推荐召回物料池的构建工作非常重要，无论我们采用规则方法或者机器学习方法（SVD、FM、RBM 等），大都是采用离线的方式进行计算的。换句话说，我们使用的是“历史数据”，当然得到的召回物料就是已经得到过充分曝光展示的商品。但是在 11.11 大促场景下，会有大量

新商品、新活动上线，甚至有些商品会频繁地上架或者下架。由于没得到充分的曝光机会，这些长尾商品无法进入推荐候选池，自然也就无法在展位上进行推荐。为了解决这个问题，我们采用了 Explore&Exploit 的方法进行处理。顾名思义，Explore 意思是探索，Exploit 就是利用得到的少部分信息预测新物料的“质量”，判断其是否值得推荐给用户，从而将其迅速加入候选池。

由于 2016 年 11.11 大促时，工作重点是个性化推荐算法的上线，在 Explore&Exploit 方面没有太多的资源可以投入，因此我们先使用了一种简单的规则方法，即首先从 长尾商品（大多数为新品）中选出一个较小的子集，按“品类|品牌|中心词”建立索引，然后在不影响个性化排序的前提下，选择流量比较大的关键展位（包括首页猜你喜欢、详情页相似推荐、详情页搭配推荐等）进行展示，收集用户对新商品的反馈数据，如果达到展示次数的阈值，该商品的 CTR 达标，就将其放入到候选物料池；反之 CTR 较低，则立即停止该商品的展示。

从反馈的结果来看，还是有相当数量的新品得到了及时地筛选，在 11.11 大促期间进行了投放展示。虽然这种方法简单直接，但需要用一部分流量 Explore，特别是一些冷门品类的长尾商品，需要更多比例的流量去尝试，一定程度上降低了大促流量的利用效率。

大促之后，为了让 E&E 机制变得更迅速、更精准，我们进行了一些新的尝试。从比较简单的 Thompson sampling、Epsilon-greedy strategy 开始，接下来又尝试了 UCB、CMAB 等算法，这些方法从实现的角度来看，是要刻画出用户对商品（或者店铺、活动等）感兴趣程度的概率分布，和上述简单直接的阈值规则方法相比，大大增强了模型的描述能力，同时也缩短了长尾商品的遴选时间，有效利用了流量。

目前我们这些 bandit 工具库已经上线，并且部署在推荐产品的全业务线上，尤其是 CMAB 相关算法会把展位位置、流量多少等场景特征也加入，主要原因是：不同展位的曝光次数、对用户的影响程度、种类限制等因素差异巨大，因此同一个商品在不同展位的曝光带来 E&E 效果是不同的。

的。

### 3.3 排序模型逐步升级

除了上文所述的召回模型，个性化推荐最重要的就是排序了，如何将召回的商品放入推荐位的槽位中，将直接影响到用户的最终选择。

国美推荐排序的“方法论”通常是指根据场景的需要，从规则排序开始，逐步上线机器学习排序方法。算法工程师也会从规则排序中先熟悉推荐场景的特点和用户反馈，在“吃掉”规则排序带来的红利之后，接下来逐步从简单的线性模型过渡到非线性模型，步步为营，不断地寻求特征层面和模型训练层面的突破。

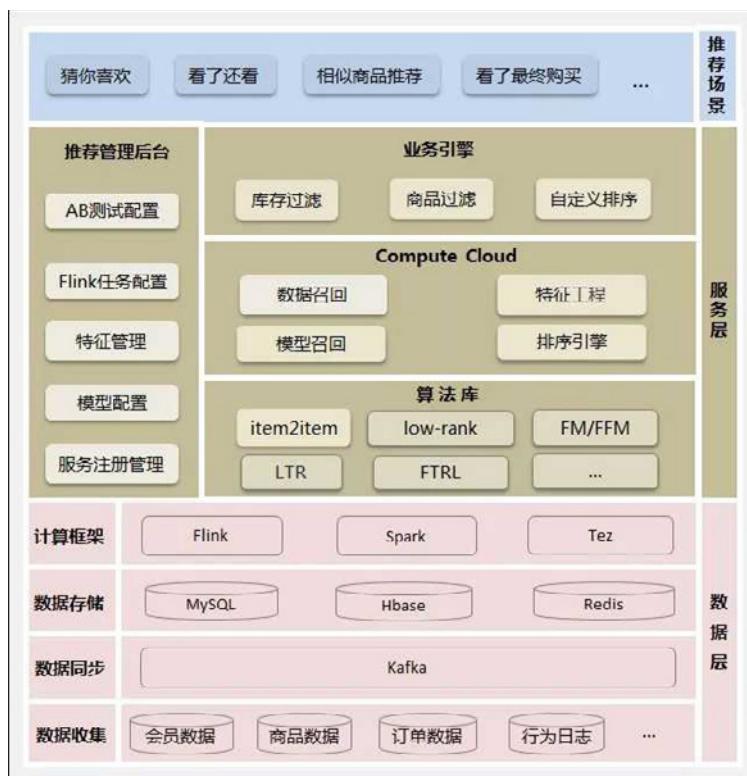
2016 年 11.11 大促我们主要使用的是点击率预估的方法进行排序，特征的整体规模较小，模型也相对简单。但是在效果上，大促期间关键展位（猜你喜欢、详情页搭配购、加购成功页等）整体 CTR 与 2015 年 11.11 同比增长 110%，CVR 提升 63%。

2016 年 11.11 大促之后，我们构建了整个排序模型流程的 Pipeline，将数据抽取、处理、标注、训练等环节进行严格意义上的流程化，让算法工程师能够准确的定位问题，专注于某一点进行优化，同时能够让 A/B 测试中各分组的逻辑界限更为明晰。

今年以来，我们将主要精力放在排序模型的优化上面，先后上线了 Mcrank、RankNet、RankSVM、FM、FFM 等方法。同时，我们在实践中尝试了用点击率预估（LR、GBDT 等）进行离线模型融合，然后再利用 Pairwise L2R 进行最终排序的方案，取得了很好的效果。在排序时还有一个问题也非常关键，比如 App 首页猜你喜欢瀑布流展位，直接将排序的结果按顺序展示，很多情况下会出现某个品类的商品“聚集”在一起，严重影响用户体验。因此我们进行了 品类穿插打散（如果考虑更多的维度进行穿插，会直接削弱排序模型的效果），满足用户对推荐结果多样性的要求。

此外，我们还在一些个性化楼层、卖场等场景下，使用了

LambdaMART 等方法。



### 3.4 在线学习初探

上面各种方法都属于离线方法的范畴，即我们通过若干时间的历史数据训练出一个模型，并假设未来一段时间内样本的概率分布不发生变化，但是实际场景中，这种假设并不成立。因此，使用实时更新的在线学习模型能够精准捕捉数据的分布变化，捕捉用户即时的购物兴趣。

今年 6 月以来，我们在架构和算法方面进行了初步尝试，上线了 BPR、FTRL 和 Online Random Forest 等模型。在 GMV、CTR 等指标方面有了显著的提升。

在线学习 的目的是为了提高排序的准确度，利用用户实时的行为数据对模型进行实时训练，使当前模型准确地反应用户当下的兴趣和倾向。但是使用在线学习也会引入新问题，比如大促期间用户的购买行为可能是非理性的，如果一味地使用在线学习将会给算法模型引入严重的偏差，因

此国美的推荐系统会定期（每天或每隔几个小时）使用离线数据进行模型训练并对在线模型进行校准。

## 四、总结

经过近几年数轮大促的洗礼，国美个性化推荐团队从场景出发，在架构、数据和算法方面都进行了深度的探索，并取得了一定的成绩。接下来，伴随着国美互联网业务的发展，用户和数据量也将快速增长，因此推荐本身的架构和算法都必须适应这种剧烈的变化。尤其在机器学习层面，下一步我们的研发重点将放在深度学习模型排序和具有人机交互功能的增强学习模型，在公司业务需求和用户个性化需求之间找到最佳的结合点。

## 作者介绍

杨骥，国美互联网大数据中心副总监。毕业于中国传媒大学并获得博士学位，博士阶段研究方向主要是计算机视觉和机器学习，包括图像的目标识别和语义分割。先后任职于凡客、京东，曾是京东PC首页与APP首页个性化推荐的开发者。多年来致力于个性化推荐系统与算法的研究和实践，目前专注于社交电商领域的深度学习技术。

# 蘑菇街 11·11：图像算法在电商大促中的应用浅析

作者 美丽联合集团



## 1. 双 11 大促的业务分析

自 2013 年转型以来，蘑菇街电商平台历经了多次双 11 大促的洗礼。通常而言，大量的商家与商品参与双 11，用户规模相比平时也会剧增。今年双 11，蘑菇街还开辟了微信小程序作为新的支点，希望以此撬动新社交电商战略。平台为用户带来价值的关键是保障商品丰富、价格合理、服务可靠。在此背景下，有很多挑战需要在复杂的业务场景中去应对，其中包括：如何提高商品管理的效率，以及如何改善用户体验。在众多的技术和产品方案中，图像算法作为一项重要能力，运用于电商场景中，支持

上述业务问题的改善。

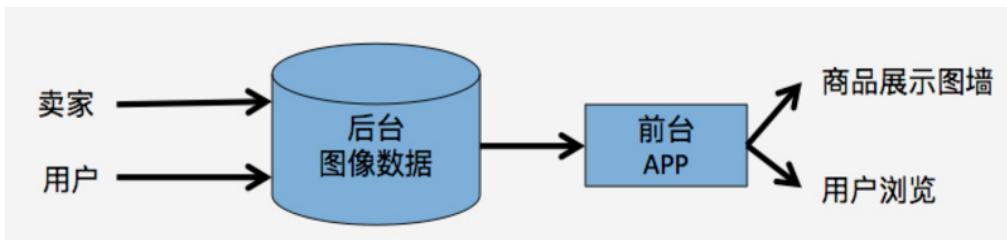


图 1. 电商中的图像数据

如同 1 所示，在电商平台中可以按照业务流向简单地描述图像数据。电商平台从商家或者用户处，获取到不同来源的图像数据，并且存放于后台图像数据库；前台 APP 产品作为面向用户的界面，基于图像数据和业务算法，把商品呈现在用户眼前，主要包括商品展示图墙页面和用户浏览页面。

蘑菇街在实践中，采用了两种类型的图像算法技术支持业务发展。第一个是图像搜索技术，用于后台商品管理和前台搜相似商品；第二个是图像标签技术，应用在商品属性管理的场景中。

## 2. 图像搜索技术的应用

### 2.1 技术原理简介

给定一个图像作为 query 输入，基于内容的图像搜索过程是在指定的图像数据库中检索，找到和 query 相同或者内容相似的图像。蘑菇街的图像搜索应用场景主要集中在商品搜索上，既有移动端的搜相似购物，也有后台运营选品的需求等。

大规模商品图像检索所面临的主要挑战包括几个方面：

1. 图像数据量大，一般电商平台的商品图像包含了主图、SKU 图、商品详情图和用户评论图等，规模达到千万至亿级别。
2. 特征维度高，图像特征是描述图像视觉信息的基础，特征表达能力直接决定了图像检索的检索精度。

3. 响应速度要快，检索系统需要具备可以快速响应用户查询的能力，一般要求检索系统能够满足实时或者准实时的要求。

针对这些挑战，蘑菇街图像搜索技术的工作主要集中在两个方面。

#### 图像特征的表达能力

随着深度学习的兴起，利用 CNN 提取图像特征，已成为图像检索领域的共识。商品图片类别众多、背景复杂，如何从丰富的图像信息中提取关键特征依然是很有挑战的问题。图像特征模块主要包含三个重要部分：数据清洗、特征模型设计、模型压缩。

利用 CNN 提取图像特征，关键在分类标签的定义。有文章 (ICLR 2017: On the Limits of Learning Representations with Label-based Supervision) 指出：模型提取特征能力的上限，不在数据集的大小，而在标签质量。因此，设计监督更强、质量高的标签，更有利于特征的表示。我们的商品标签有两个来源，一个是商品在类目体系中从属的类别，另一个是商家对商品的描述。数据清洗过程主要解决商家打标的标签和图像实际内容不符合的问题。利用自动化图像标签模块，可对商品图片自动打标，辅之以人工矫正。通过这种方式我们累积了数以千万计的样本图像数据，所涉及的标签 label 数目有几千种，从而构建了高质量的训练样本。

特征模型的设计以 ResNet(残差网络) 为基础，根据 ResNet 是浅层网络集成学习的思想 (NIPS 2016: Residual Networks Behave Like Ensembles of Relatively Shallow Networks)，我们通过设计不同尺度卷积核并拼接 (Concat) 在一起，提高了浅层网络的表达能力；同时适当控制深度，并改进 ResNet 中影响优化的 Shortcut 结构。试验证明网络的改进是有效的，改进后的网络在实际数据集合上的 top1 accuracy 是 61.8%，而传统的 ResNet-50 是 56.6%。

特征模型部署在 GPU 服务器上，为控制系统的整体响应时间，需要缩短特征提取的时间，因此要对深度学习网络模型进行压缩。压缩算法采用的是 (ICLR 2017: Pruning Filters for Efficient ConvNets) 所提到的剪枝策略。具体的做法是：针对每个卷积核计算其绝对值和，然后排序，针

对绝对值小的权值和通道进行剪枝。流程中包括两个主要步骤：首先按照一定比例（比如 10%）进行压缩，然后进行模型的 fine-tunning 训练；两者交替迭代进行，直至模型精度的下降超过预设的目标，流程结束。最终我们所获得的特征模型在 GPU 卡 K40 上，单次特征抽取的时间在 40ms 内。

### 近似最近邻查找

鉴于搜索数据库数据量级很大，对每个查询都要计算所有的距离是非常困难的，同时存储数千万图片的高维残差网络特征向量需要耗费巨大的存储空间。为了解决这些问题，采用了近似最近邻算法中的局部优化的乘积量化算法（Product Quantization，PQ），训练得到粗量化质心和细量化质心，粗量化的结果用来建立倒排索引，细量化结果用来计算近似距离。通过这种方法，既能保证图像索引结果的存储需求合理，也能使检索质量和速度达到更好的水平。

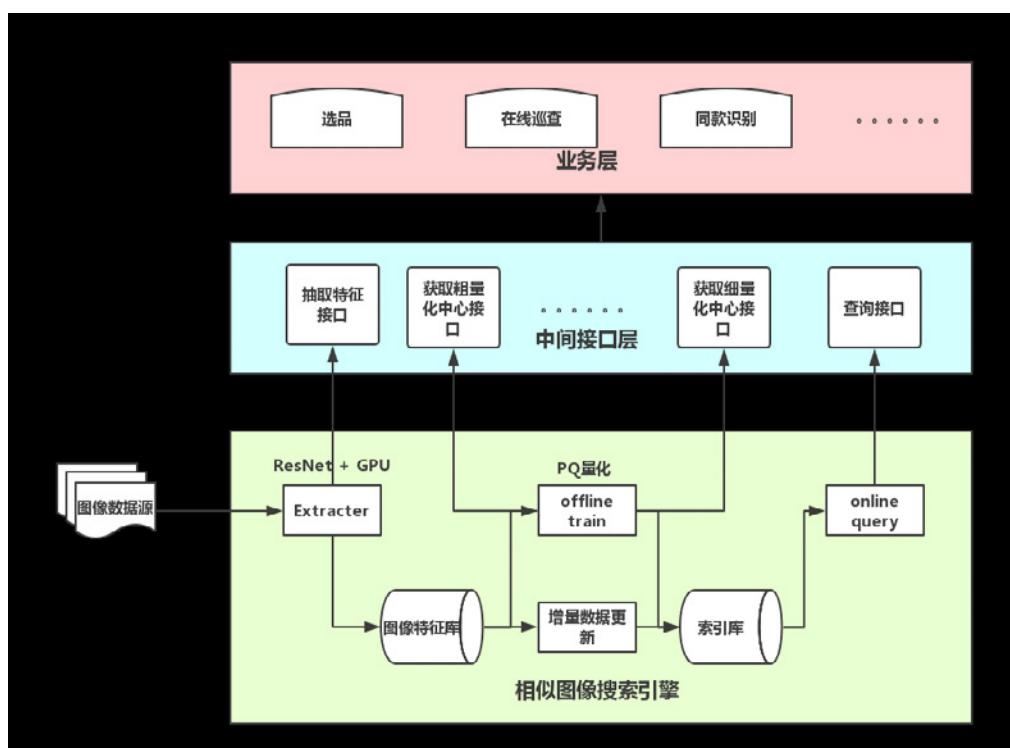


图 2. 图像检索的系统架构

图像检索系统的整体架构如图 2 所示。基于底层的图像搜索算法，通过中间接口层提供给具体的业务使用，提升了相似图像搜索的扩展性，能够快速地响应实际的需求和应用。

## 2.2 应用 1：同款商品识别

电商基础业务中，需要审核商家上传的商品图片。我们在实践中基于相似图像搜索技术，构建了同图识别系统。系统产出了全量图像数据的索引库，基于相似搜索引擎来查询商家图片是否为商品库中的相同图像。根据查询结果，结合业务规则，判断商品是否为同款。图 3 给出了同款商品识别的系统概要图。

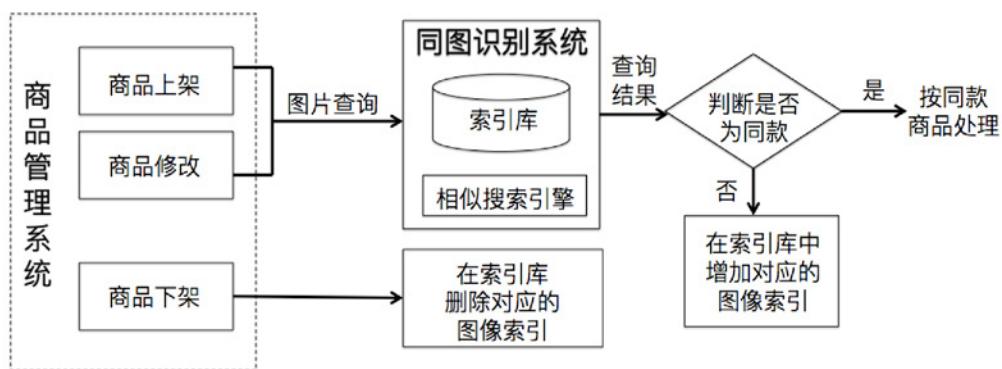


图 3. 结合图像信息的同款商品识别

目前该系统部署在蘑菇街电商平台中，提升了商品管理的效率。在亿级图像索引规模下，系统识别准确率为 99.06%，单张图像查询的整体响应时间为 20ms。

## 2.3 应用 2：移动端搜相似商品

蘑菇街 APP 上提供了搜相似商品的功能。如图 4 所示，其过程是点击单个商品图像右下角的搜索图标，将与该图像相似的同类商品展现给用户。

图 5 展示了系统概要图。在实现过程中，采用了图像搜索技术来承担相似图像查询，从而召回相似商品列表，然后结合业务因素和图像相似性，进行商品排序。通过该功能，能够提升用户在蘑菇街 APP 上的浏览

体验，有利于发现更多相似商品；同时，用户的停留时长也有所增加。



图 4. 搜相似的用户界面，左 (a) 商品展示原图，右 (b) 相似商品列表页面

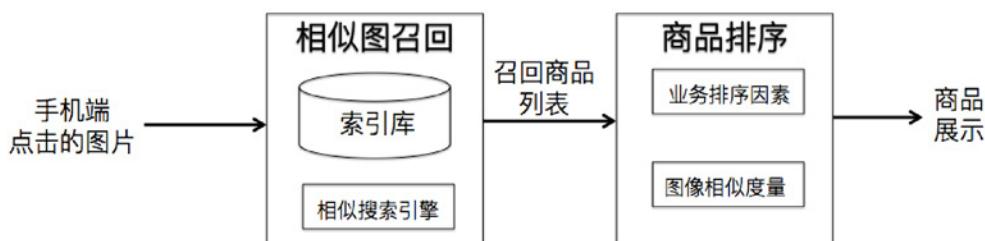


图 5. 搜相似商品的系统概要图

### 3. 图像标签技术应用

#### 3.1 技术原理介绍

图像标签技术的任务是通过图像算法，自动识别图片内容，如场景、风格、主体名称、颜色、图案等。通常来讲，在电商中对商品图片做图像标签的处理流程如图 6 所示，其中所涉及到的技术模块简述如图 6 所示。

##### 区域提取

这个阶段利用图像分割技术，将商品主体从背景中分离出来。以服装图像为例，区域提取的目标是对图片进行精细化语义分割，排除背景与服



图 6. 图像标签处理流程

装、服装与服装之间的相互干扰。针对服装模特图片，我们通过 Human Parsing 算法，把主要区域提取出来，例如：头肩、上衣、裤子、鞋、包包等。

图像语义分割是图像理解的基础技术，在服饰信息分析、自动驾驶系统（具体为街景识别与理解）、无人机应用（着陆点判断）以及穿戴式设备应用中举足轻重。众所周知，图像是由像素组成，语义分割就是将像素按照图像中表达语义含义的不同进行分组和分割。如图 6 所示，紫色区域表示语义为“上衣”的图像像素区域，荧光蓝代表“下装”的语义区域，军绿色表示“包包”，橙色则表示“鞋子”区域。在图像语义分割任务中，输入为一张  $H \times W \times 3$  的三通道彩色图像，输出则是对应的一个  $H \times W$  矩阵，矩阵的每一个元素表明了原图中对应位置像素所表示的语义类别（Semantic label）。因此，图像语义分割也称为“图像语义标注”。

在语义分割领域，全卷积网络 (Fully Convolutional Networks, FCN) 推广了原有的 CNN 结构，在不带有全连接层的情况下能进行密集预测。FCN 使得分割图谱可以生成任意大小的图像，且与图像块分类方法相比提高了处理速度。实际上几乎所有关于语义分割的最新研究都采用了

FCN 结构；不过该框架中的池化层在增大上层卷积核的感受野、聚合背景的同时，却丢弃了部分位置结构信息。

在服装语义分割中，人体结构和服装位置之间的相对关系，对提高分割效果来说至关重要。我们在实际工作中，借鉴了 DeepLab 论文 (DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. arXiv:1606.00915, 2016) 的空洞卷积 (Atrous Convolution) 思想，保留了池化层中所舍弃的位置结构信息。

丢失的位置结构信息主要由于重复池化和下采样造成，因此我们的网络中移除了最后的若干个最大池化层下采样操作，并对滤波器进行上采样，在非零的滤波器值之间加入空洞，进行空洞卷积。如图 7 所示，(a) 中的常规卷积只能获取到较小感受野上的稀疏特征；(b) 中方法采用空洞卷积后可以得到较大感受野对应的更丰富特征，对应到服装语义分割也就保留了人体和服装之间的位置结构信息，有助于服装分割效果的提升。

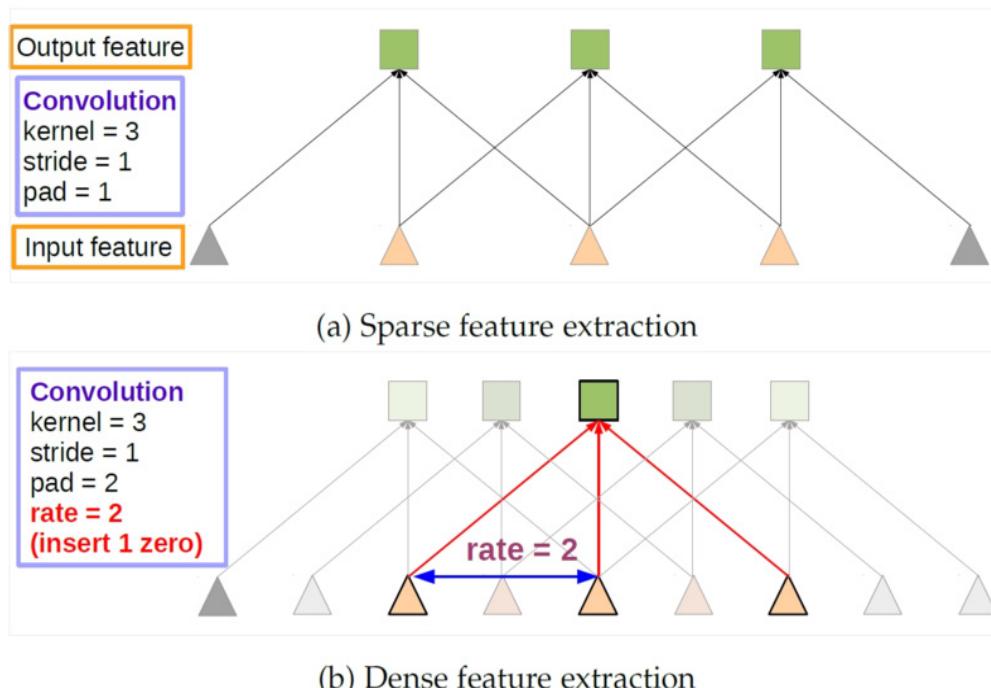


图 7. 图像语义分割中的卷积操作

## 属性分类

根据电商业务特点，主要从三个不同的维度来定义标签体系，包括类目、元素、颜色。类目主要描述商品是什么，元素和颜色体现商品有什么样的性质。

我们定义的类目包括服装类、生活类、化妆品，元素范围包括风格、纹理、版型等。以服装为例，标签信息比较复杂，覆盖到服装的多级类目、颜色、领型、衣长等，这导致同一张图存在多个标签。在实际工作中我们聚焦于以下两个方面的属性分类解决方案。

### 标签的层级关系建模。

电商商品的层级类目结构导致一件商品具有层级化的标签，例如 T 恤：它的一级类目是服装，二级类目是上衣，三级类目是 T 恤，因此一件 T 恤的商品图单类目标签就有 3 个。根据不同标签和相应的数据训练单独获得 3 个模型是一个可行方法，但是系统复杂度过高。

我们采用多级 word tree 结构来实现一个整合多套标签的模型，同时用标签间的关系来约束输出，概率表示为： $P(T \text{ 恤}) = P(T \text{ 恤} | \text{上衣}) * P(\text{上衣})$ 。根据这个思路设计，用一个模型就能表示出层级关系的多套标签，不仅充分利用了同一批数据，而且通过条件约束之后输出的标签更精准。

### 标签的多维度特性。

对于同一件商品会有不同维度的信息描述，以 T 恤为例，它有衣长、领型、袖型等维度的信息。因此对于多维度的标签模型，我们采用多任务（multi-task）的方法，用一个网络同时提取不同维度的图像特征，能够统一描述图片内容。

在实际系统中，我们利用了百万级别的样本进行模型训练，基础模型的网络结构是残差 18 层网络（ResNet-18）。业务测试中类目的准确率是 92.46%，元素分类的准确率是 91.10%。

## 颜色量化分析

用色彩来装饰自身是人类的原始本能，色彩在服饰审美中有着举足轻重的地位，自古至今都是服装三大要素之一，因此服装颜色标签识别是重

要部分。通过区域提取过程获得上衣、裤子等服装单品的区域后，对单品图像采用改进的 Mean-Shift 算法进行颜色聚类，得到服装单品的主要颜色占比。

实际应用中，蘑菇街商品图片的颜色会受到拍摄光线和滤镜处理的影响，这给我们的颜色识别带来了挑战，主要表现为两个方面：

颜色空间的选择：不同的颜色空间有不同的特点，Lab 色彩空间具有单独的亮度通道，因此为了减小光照对聚类算法的影响、提升颜色聚类的准确度，我们会在 Lab 色彩空间对图片色彩进行一定的矫正，再进行颜色聚类，以尽可能减少光照和滤镜对颜色识别的影响。

服装色卡定义：根据蘑菇街服饰颜色分布及商品标签需要，定义了蘑菇街标准的 72 色服装色卡。基于颜色聚类的结果，获取到主体颜色的聚类中心信息，所占比例最大的聚类中心所对应的色卡，被定义为最终输出颜色名称。

### 3.2 应用：商品属性的自动填写

商家在发布新品时，需要填写商品的标题、上传图片，填写商品的属性值，以及详情页信息。当上新量很多的时候，特别是筹备双 11 期间，填写商品信息比较费时，加大了商家的工作量。而图像算法能够在商家上新的环节，通过分析上传的图片，得到图中的关键信息，为商家提供便利。

以服装类目举例，商家上传了商品图片后，我们通过图像标签技术模块，计算得到图中商品的一系列属性信息。例如图 8 所示，这些信息包括：类目（毛呢外套）、袖长（长袖）、版型（收腰）、领型（西装领）、衣长（长款）、风格（韩系）、颜色（藕粉色）等。利用这些信息，自动帮商家填写好对应的属性，节省了商家选择属性值的时间。当商家发现图像算法识别错误时，可以在自动填写的基础上，对已填写内容进行手动修改。整个流程能够大幅度减少商家上新填写信息所需时间，提升商家的业务效率。



图 8. 商品属性自动填写

## 4. 结语

在实际的业务场景中，图像算法开发是基于应用来驱动的，为保障平台运营和用户体验提供价值。我们的工作，通过图像搜索技术可以自动识别平台上的同款商品，提升后台商品管理的效率；也能够帮助用户发现更多相似商品，改善用户体验。同时，运用图像标签技术，为商家发布新品节省信息填写时间，提升了商家效率。

我们在日常的开发过程中积累图像算法的基础模块，并在双 11 的业务开发中拓展其运用场景；未来将根据业务中的数据变化、场景变化，进行技术迭代开发，从而为不断升级的业务需求提供保障。

致谢：本文工作是蘑菇街图像算法组同学的共同贡献，特此致谢！

# 苏宁 11·11：从 0 到 1，苏宁 API 网关的演进之路

作者 何 翔



2012 年，在开放云融推动各产业全面发展的大背景下，苏宁 API 对外开放。基于苏宁各内部业务系统的资源，开放丰富的 API 服务，提供给苏宁商家、供应商、售后服务商、物流公司、软件服务商等合作伙伴所需的数据和信息。实现外部系统与苏宁的完美对接，使业务的处理更加高效、便捷。

通过阅读该文章你会了解到如下信息：

1. 一切皆基于标准
2. 系统重构详解

3. 高可用设计
4. 监控之道
5. 应对 O2O 购物节

到现在为止，苏宁已开放了平台业务、自营业务、自采业务、O2O 业务、政企业务、特卖业务、4PS 业务等几百个 API 接口，每天承载海量的 API 接口调用。

## 一、一切皆基于标准

俗话说“无标准不平台”，苏宁 API 设计之初，没认识到标准的重要性，遇到过很多问题，走过很多弯路，接下来细说下苏宁 API 的标准化设计的过程。

### 1. 从接口名开始

谈起 API 接口，首先会想到接口名（接口英文名）。

先来看一组接口名：

```
suning.order.get  
sunin.selfmarket.order1.query  
sunin.custom.book.item.query  
sunin.retuenBadArticleHandleResults.add  
sunin.shoppingmallsalesdata.saveandupdate
```

以上接口名有如下问题：

- 长度不一。
- 风格不一致。

而一个标准的接口名应该有如下特征：

- 风格统一（都为英文小写）
- 模式固定（xx.xx.xx.xx）
- 简洁明了
- 有扩展性
- 分类与分层

举例：suning.custom.order.get

格式：suning. 业务分类. 模块简称. 操作符

## 2. 文档规范化

用户通过阅读 API 文档来使用 API 接口。一个规范的 API 文档有助于用户理解 API 的用途，掌握 API 的使用。

### API 文档组成



### API 文档详情

#### `suning.custom.order.note.get` 订单备注查询

通过此接口可查询订单备注。  
1. 订单号可通过接口“`suning.custom.order.query`”、`“suning.custom.historyorder.query”`、`“suning.custom.ordercode.query”` 获取；

公共参数				
orderCode	String	是否必须	示例值	描述
		Y	4511680451	订单号

请求参数				
名称	类型	是否必须	示例值	描述
orderCode	String	Y	0000020161398	订单号

响应参数				
名称	类型	示例值	描述	
orderCode	String	0000020161398	订单号	
noteContent	String	轻拿轻放	备注内容	
noteFlag	String		交易备注旗帜	

- 公共参数
- 请求参数
- 响应参数
- 请求示例
- 响应示例
- 异常示例
- 业务异常码
- 公共异常码
- API 工具
- FAQ

**请求示例**

**响应示例**

**异常示例**

**业务异常码**

**公共异常码**

**API工具**

请求示例、响应示例、异常示例、业务异常码、公共异常码、API工具、FAQ

### 3. 协议主流化

苏宁 API 主要是基于 Http(s)+OAuth2.0 协议实现的。可以使用 xml 或者 json 格式进行报文传输。并提供主流 (Java、.Net、PHP、Python) 四种语言 SDK 方便用户使用。

API 接口为一个全局统一的入口：

<https://open.suning.com/api/http/sopRequest>

### 4. 服务契约固定化

这里服务契约是指苏宁 API 系统与苏宁内部系统之间的服务契约。苏宁 API 承接着苏宁内部多方系统。如果不固定服务契约，苏宁 API 就无法做到标准化，这是一个 API 标准化的一个前提。有了服务契约的固定，才能实现动态发布 API，缩短 API 的研发周期。

### 5. 异常码标准化

## 什么是异常码

即调用 API 接口异常场景下返回的错误码。苏宁的 API 接口返回异常码的同时也会返回中文详细描述，以增强异常码的可读性。异常码通常表示某种异常场景，具有唯一指向性。

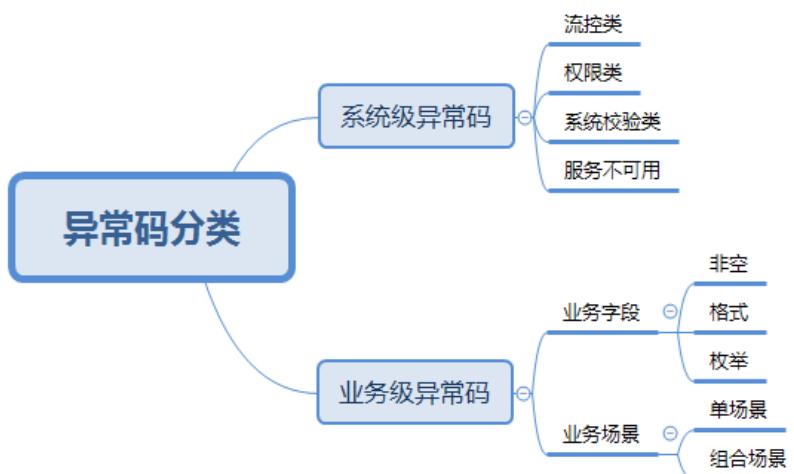
## 异常码的设计问题

异常码常见的问题包括：无统一风格、无分类、无唯一性、难以理解等。基于以上问题，苏宁的 API 网关在异常码方面进行一些标准化设计，使异常码变得有如下特征。

## 改进后的异常码特征

- 唯一性
- 风格一致
- 有层级有分类
- 可扩展

## 异常码的分类



## 异常码标准化带来的意义

### 1. 异常码监控

每个异常码都代表不同的异常场景，异常也有主次之分，重要的异常场景可通过异常码配置监控。

## 2. 接口告警

基于异常码监控，一个接口可配置多个异常码监控告警。

## 3. 定位问题

异常码具有唯一指向性，能快速定位问题。

## 4. 解决问题

在基于异常码具有唯一指向性的前提下，能快速解决问题。

# 6. API 接口配置化

在以上标准化的基础上，为了缩短 API 接口上线周期，我们实现了 API 接口配置化，能支持自动发布 API 接口、动态修改配置项。具体界面如下图：

## 基本信息配置

The screenshot shows the 'Basic Information Configuration' step of a four-step API configuration process. The steps are: 1. 基本信息配置 (highlighted in blue), 2. 外部请求API, 3. API返回外部, 4. API请求内部. The configuration form contains the following fields:

- 系统名称: [redacted]
- 菜单目录: 定制接口 (selected) 平台业务 交易API
- 接口名称: suning.custom.ordernote.modify
- 接口别名: [redacted]
- 接口中文名称: 订单备注修改 (highlighted in green)
- 是否分页: 不分页
- 每页条数上限: 请选择

## 外部请求 API

The screenshot shows the 'External Request API' configuration step of the four-step process. The steps are: 1. 基本信息配置, 2. 外部请求API (highlighted in blue), 3. API返回外部, 4. API请求内部. The configuration form contains the following sections:

- 入参配置:**
  - 请求参数配置:
    - orderNote
      - colorMarkFlag
      - noteContent
      - noteFlag
      - orderCode- 示例按钮:**

示例正文: [查看示例正文](#)

[ 上一步 ] [ 下一步 ]

## API 请求内部

The screenshot shows a step-by-step configuration process for an API. Step 3, 'API调用外部' (Call External API), is highlighted in blue. The interface includes:

- Response Configuration:** Shows return parameter configurations: 'orderNote' and 'result'.
- Example Text:** A link to view example text.
- Exception Handling:** A table for abnormal handling settings, showing one entry for 'biz custom modifyordernote missing paramet...' with '参数必须' (Parameter Required) and '请检查此参数的值 \*号表示参数名字命名' (Check the value of this parameter, \* indicates parameter name) under '解决方案' (Solution).

## 7. SDK 自动化

当 API 接口发布上线后，会自动生成（Java、.Net、PHP、Python）四种语言的 SDK，并执行自动化测试，测试通过后上传官网，供外部用户使用。

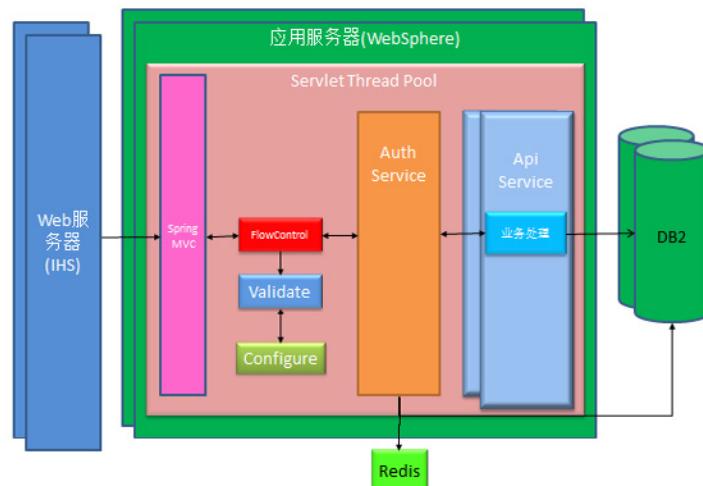
## 二、系统重构详解

随着 API 接口数量越来越多，一些问题逐渐暴露出来，下面详细介绍几个关键问题的重构详解。

### 1. 响应时间慢

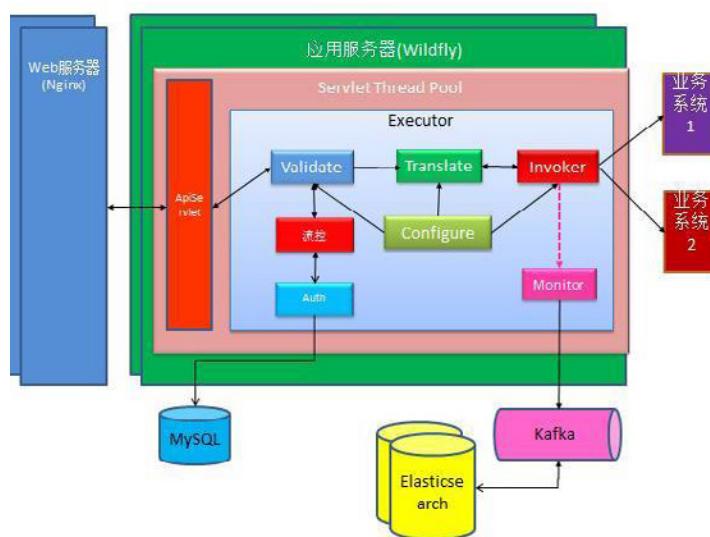
系统上线之初，选用 IBM 的 IHS 和 Websphere、DB2 来部署 API 系统，系统架构如下图。

这套架构的优点是简单，整个 API 运行在 Servlet 容器中，API 全局入口是一个 Spring MVC 的 Controller。缺点是 API 系统不是通过服务调用业务系统，而是自己要连接业务数据库，并处理业务逻辑。一个 API 接口对应一个 API Service，实现了简单的流控、校验、权限认证、本地配置功能。随着时间变迁，API 接口数量越来越多，API 接口开发周期变长、响应时间逐渐变慢了。



## 2. 并发支持低

为了解决 API 接口响应时间慢的问题，对系统进行如下重构：

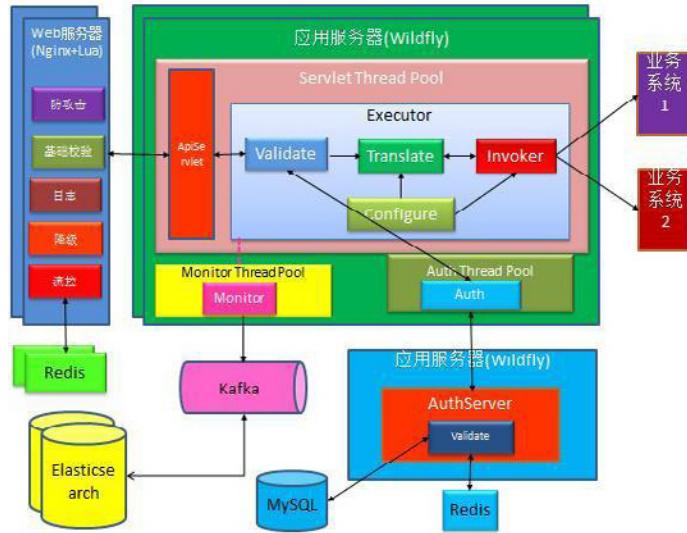


此时 API 系统架构变成 Nginx+Wildfly(JBoss)，整个 API 运行在 Servlet 容器中，API 全局入口是一个 Servlet；同时新增了流控、监控、调用器组件；API 业务逻辑处理修改为通过 Hessian 方式调用业务系统；监控组件同步发送日志到 kafka，Elasticsearch 从 kafka 消费日志，建立索引。

经过上面系统重构后，系统运行一段时间，发现接口响应返回越来越慢，很多无效调用和攻击影响正常的请求，同时核心链路并发支持低，也是影响接口响应慢的原因。

### 3. 接口分流

为了解决无效调用、安全攻击、核心并发支持低导致 API 接口响应时间慢的问题，对系统进行如下重构：



此次系统重构主要是 强化接入层的功能，在接入层实现了防攻击、基础校验、日志、降级、流控等功能，同时将核心服务权限认证进行平台化，增加核心链路的并发度和增强系统扩展性，将监控组件异步化。

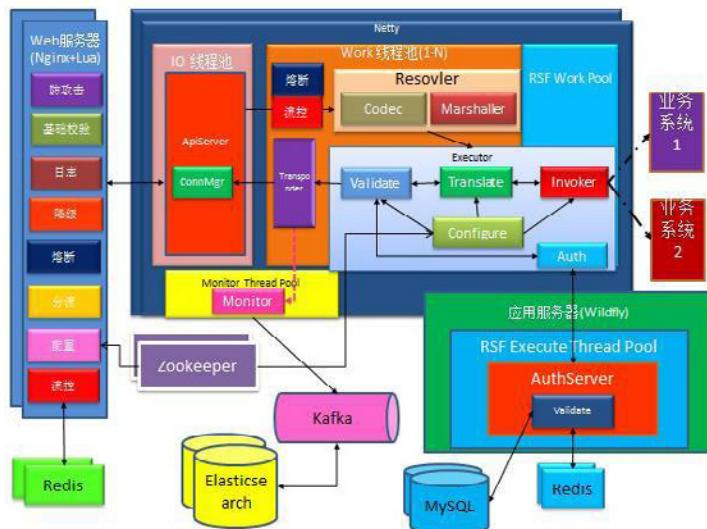
运行一段时间后，又出现了一个现象，运行快的 API 接口会被运行慢的接口影响，也变得运行慢。

### 4. 核心组件重构

为了解决接口之间互相影响的问题，对系统进行如下重构。

在接入层进行接口分流，不同的接口请求到不同的后端服务，这样接口进行应用层的资源隔离；并对应用层进行重构，重写核心组件；整个应用层基于 Netty 提供 API 服务；IO 线程负责连接的接收与建立；Work 线程负责 IO 的读写；同时支持多组 work 线程池，用来隔离不同接口和业务直接的互相影响；服务调用组件修改为苏宁自研分布式服务框架 (RSF)，使用异步回调方式，提高系统的吞吐量，核心权限认证服务，也修改 RSF 调用方式，提高服务的性能，新增 Zookeeper 来提供配置管理，来实现分

流的动态配置修改。

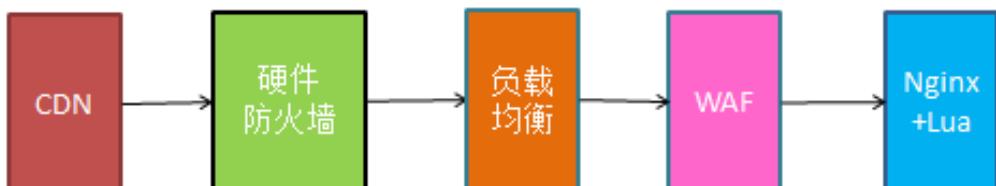


### 三、高可用设计

在系统重构过程中，除了提高系统的性能，系统的高可用设计也尤其重要。下面将详细介绍苏宁 API 的高可用设计包括哪些内容。

#### 1. 安全保障

一次 API 接口请求链路如下，每个链接职责分离。

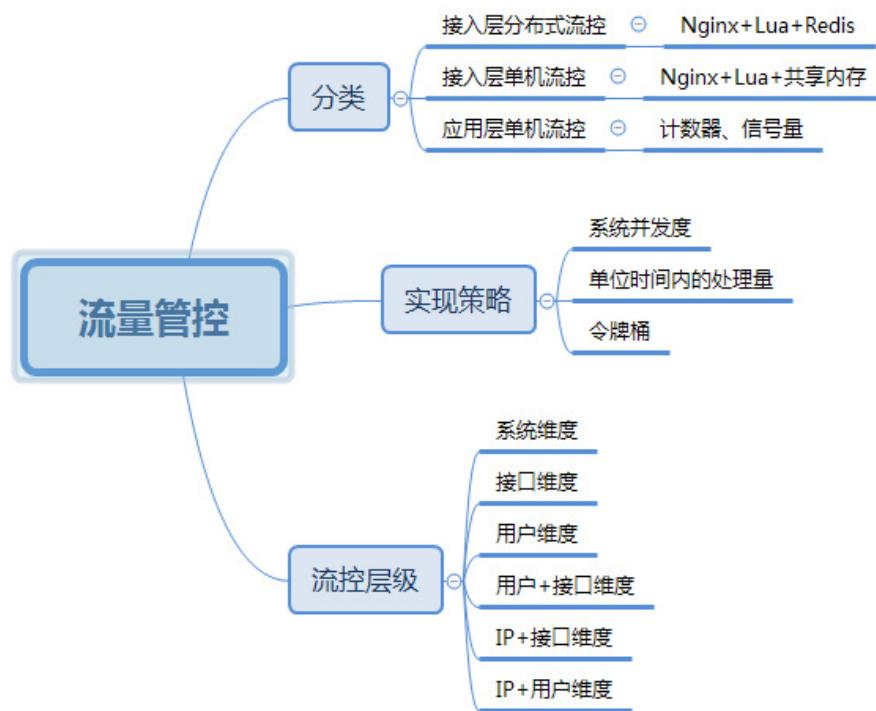


- **CDN:** 能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上
- **硬件防火墙:** 重点关照外联链路和外联 VPN 做了一些通用的防火墙策略
- **负载均衡:** 能支持域名为维度做并发控制，限制每秒的最大请求数

- WAF：基于 Nginx+Lua 实现，提供同样的安全防护和域名维度、URL 维度、IP+URL 维度的准实时流量管控。
- Nginx+Lua，也是基于 Nginx+Lua 实现，提供 API 接口维度的安全攻击防护和多层次的流控策略，来保证系统的稳定性。

## 2. 流量管控

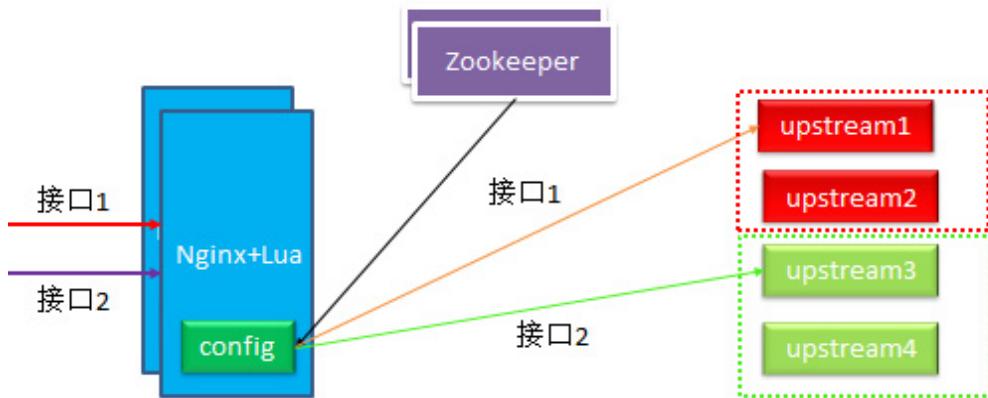
API 系统流量管控，主要分为接入层和应用层流控。接入层主要是基于 Nginx+Lua 来实现的，应用层基于计数器和信号量等实现。具体信息如下：



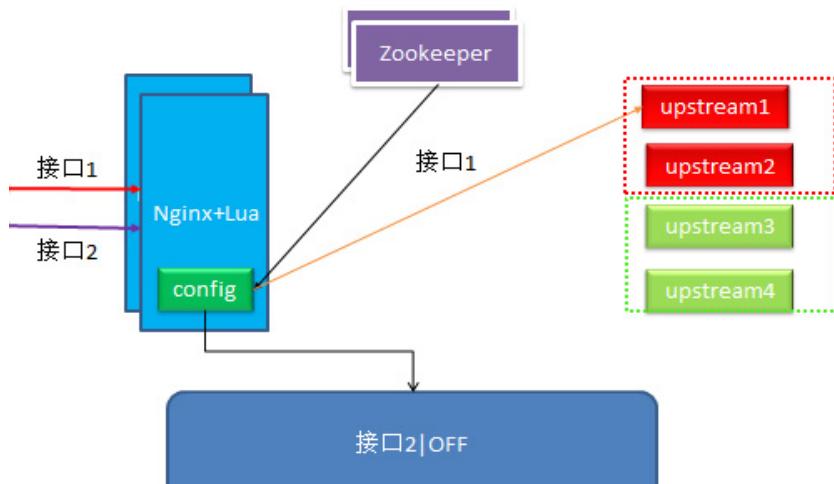
## 3. 接口分流

接口分流基于 Nginx+Lua 实现，预先对每一个 API 接口进行逻辑执行区域的分类，不同的分类对应不同的后端服务，通过 Zookeeper 动态管理配置项，一个接口请求会找到对应 upstream Server。

## 4. 服务降级



接口降级同样基于 Nginx+Lua 实现，Nginx 共享内存中存放接口的降级信息，通过 Zookeeper 动态管理配置项，如果接口被降级就会直接返回异常码，否则接口就会执行接口分流逻辑，请求会找到对应 upstream Server。

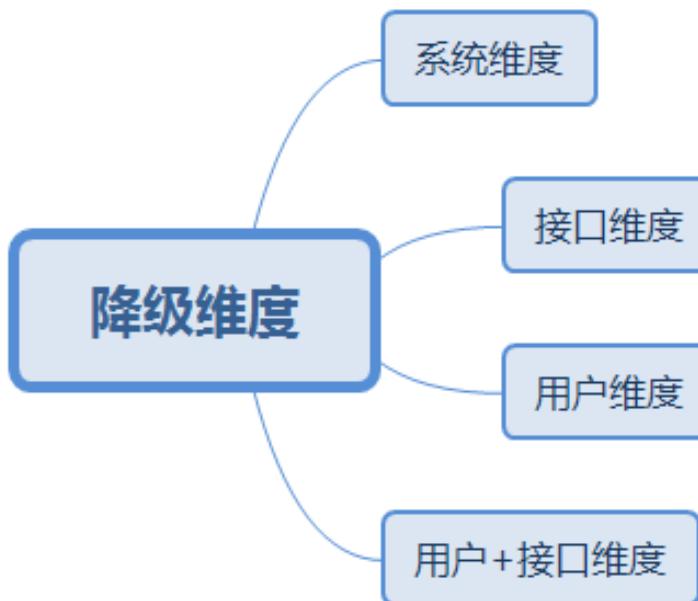


同时服务降级也支持多个层级，具体如下图所示。

## 四、监控之道

当 API 接口数量和调用量达到一定程度后，面临了如下问题：

- 日志监控
- 如何记录接口完整调用记录？
- 用什么存储接口调用记录？



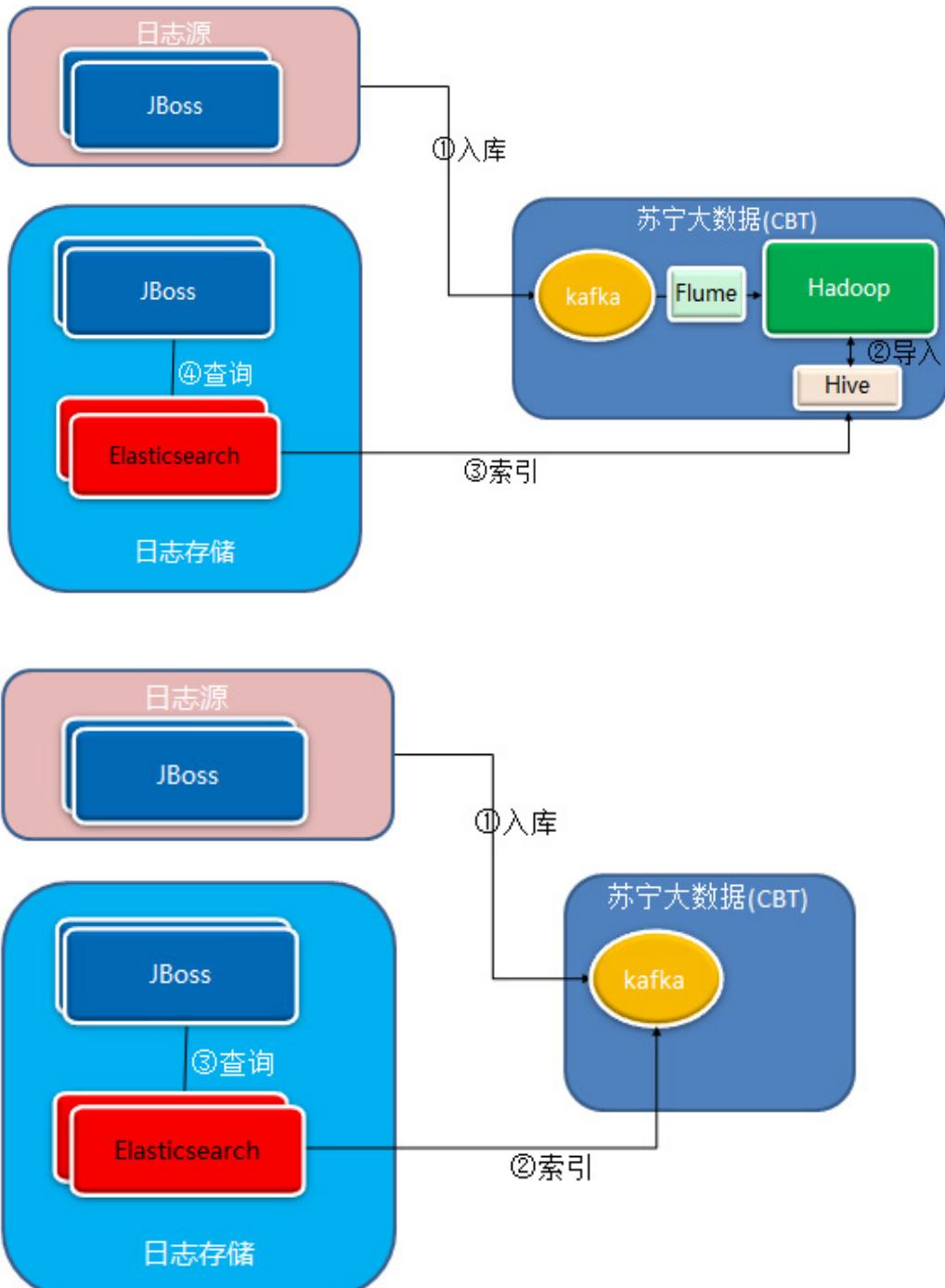
- 如何准确查询接口调用记录?
- 查询日志能否准实时?
- 接口统计
- 如何实现接口调用统计?
- 是离线统计还是实时统计?
- 按照哪种维度进行统计?
- 如何存储统计数据?
- 如何展示统计数据?

## 1. 日志明细

第一版日志明细实现系统设计如下所示。上线后，由于 Flume 采集任务和 Hadoop 任务导致日志明细不能实时查看，整个日志延迟 20 分钟。基于上述原因对系统进行重构，结果如下图所示。

去掉 Flume 采集任务和 Hadoop，直接采用 Elasticsearch 消费 kafka 数据，实时建索引。

日志源和 kafka、Elasticsearch 都具备扩展性，满足性能要求。



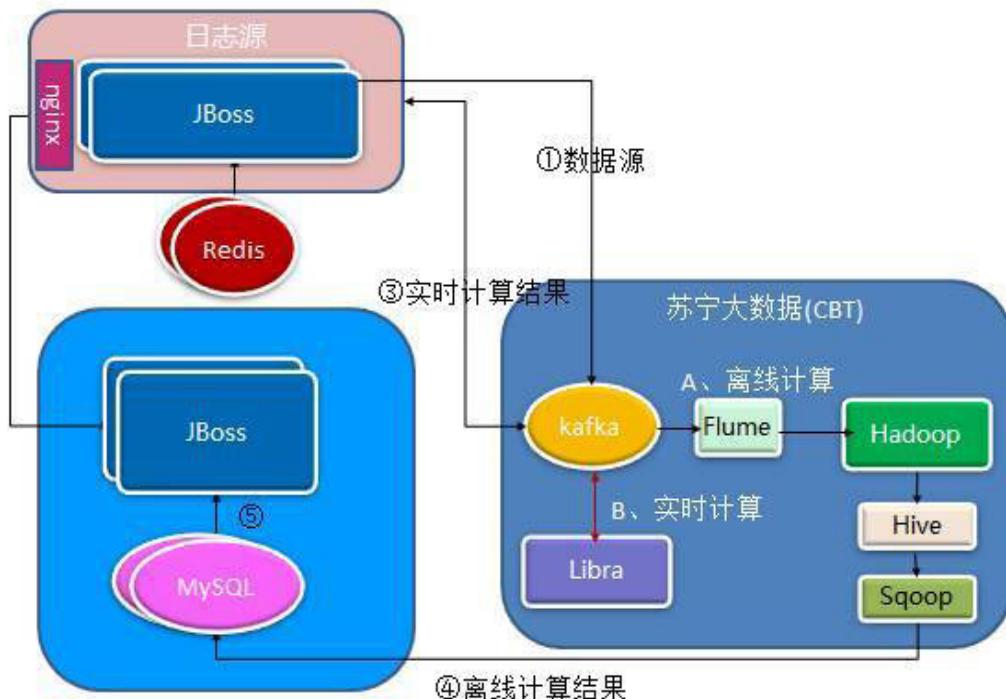
## 2. 接口统计

依赖于苏宁数据大数据平台来完成接口的统计。

使用 Hadoop 进行离线统计，根据不同维度编写 Hive 语句转换为

MapReduce 任务进行计算，通过 Sqoop 将计算结果导入到 MySQL 数据库中。

使用 Libra 实时计算平台进行实时统计，根据不同维度编写 epl 语句转换为实时计算任务进行计算，计算结果发送到 Kafka，应用监听 Kafka 消费实时计算结果，存储到 Redis 中，并设置数据过期时间。



Libra 实时计算平台 是苏宁在 Storm 的基础上二次开发封装的，通过编写 EPL 计算语句 (EPL 是 esper 中一种类似 SQL 的语言)，转换为一个个计算规则，提供实时计算功能的平台，主要特点为配置简单能够实时计算需求的快速上线、支持分布式计算、支持动态扩容，以满足大数据量的计算需求。

### 3. 监控的意义

在没有监控之前，存在如下问题：

- 定位问题很困难
- 没有接口运行数据，无法优化接口性能

- 没有接口统计数据，无法考核服务质量
- 而有了监控之后：
- 方便定位问题
- 优化性能隐患
- 方便考核服务质量
- 方便实现业务扩展

#### 4. 智能告警

苏宁有如下告警平台和告警机制，能够准确地通知开发及运维人员系统的异常状况，有助于快速定位问题和解决问题。

ZABBIX 监控系统。全方位监控操作系统、应用系统层面（ZABBIX 是一个基于 WEB 界面的提供分布式系统监视以及网络监视功能的企业级的开源解决方案）。主要监控项包括 CPU、系统负载、内存、网卡流量、磁盘使用量、连接数、宕机检测等。告警方式为短信或者邮件。

穆加。苏宁智能告警平台，整合了 ZABBIX 监控系统、云迹服务端性能监控、Kafka 后台管理系统、决策分析平台、机器学习平台、准实时计算平台监控来源，提供更细致化的、丰富的指标种类，提供豆芽（苏宁自研的实时通讯工具）通知、短信、邮件、语言等多种告警方式。

云迹异常监管系统。能针对某个系统的某种异常，在单位时间内异常量或环比增长量超过配置的阀值，则进行短信或者邮件告警。

以上都是苏宁系统提供的通用告警机制，接下来介绍苏宁 API 系统实现的告警机制：

1. 提供了基于 API 接口维度的告警，调用时长、调用次数、失败次数、失败率等维度，在单位时间内超过配置的阀值，则进行短信或者邮件告警。
2. 提供了基于 API 接口异常码维度的告警。在单位时间内异常量或环比增长量超过配置的阀值，则进行短信或者邮件告警。

## 五、苏宁 API 网关应对 O2O 购物节

从 2012 年开始，每年双十一苏宁都带给消费者一次次购物享受之旅。今年苏宁易购双十一的主题是“不止所见嗨购 11 天”，换句话说，这场狂欢盛宴将始于购物，但不止于购物，为消费者提供其他电商没法实现的“任性”。

O2O 购物节是一次狂欢盛宴，对每一个苏宁人来说是一场必胜的战役，为了保证 O2O 购物节系统正常，需提前做好系统保障工作，正所谓不打无准备的战役。

O2O 购物节中苏宁 API 网关提供哪些应用场景辅助商家、供应商、服务商进行商品交易？下面列举了主要的应用场景：

- 商品发布

获取苏宁采购目录

通过接口 suning.custom.category.query 获取 categoryCode

获取商品品牌信息

通过接口 suning.custom.brand.query 获取 brandCode

获取苏宁产品库信息

通过接口 suning.custom.product.query 获取苏宁产品信息

获取苏宁产品详情信息

调用接口 suning.custom.productdetail.query

商品内容维护

调用接口 suning.custom.itemcontents.add

获取商品参数模板

调用接口 suning.custom.itemparameters.query

产品申请

调用接口 suning.custom.item.add

- 订单同步

获取三个月前的订单

调用接口 suning.custom.historyorder.query 批量获取订单（三个月前的历史订单）

## 获取三个月内的订单

调用接口 suning.custom.order.query 批量获取订单（三个月内的订单）

调用接口 suning.custom.ordercode.query 批量获取订单号

调用接口 suning.custom.order.get 查询单笔订单

## 获取增量订单

调用接口 suning.custom.ord.query 根据订单修改时间批量查询订单信息

- 物流发货

苏宁物流或第三方物流配送

调用 suning.custom.orderdelivery.add 对订单进行发货。

商家自配送

调用 suning.custom.orderselfdist.add 对订单进行发货。

采用该接口发货时，需要填写商户自己执行配送的人员的姓名和手机号。

## 战前

- 容量评估

根据 O2O 购物节订单预估量，对系统进行容量评估。分场景压测系统得到性能测试结果，再根据结果进行性能达标评估。不达标的系统需进行自动扩容，扩容后需对系统进行性能测试，检验系统容量是否达标。

- 性能测试

每次战前，各个系统都要提前对关键路径进行性能测试，找出性能瓶颈并优化。

- 预案梳理

梳理系统的紧急预案，在 O2O 购物节中异常事件发生时进行预案处理，如服务降级，关闭非核心功能，保证核心功能。

- 预案演练

根据 O2O 购物节中异常事件启动对应的紧急预案，看看是否能达到

预期效果。

## 战时

- 集中值班

O2O 购物节期间安排 24 小时集中值班，来监控系统运行情况，应对突发紧急问题，并解决问题。

- 系统监控
- 监控系统运行情况，各项指标 (CPU、IO、内存、网卡流量等)
- 苏宁云迹链路监控大盘



- 预案执行

O2O 购物节期间发生的异常事件，根据战前准备的预案执行，监控执行结果。并更新预案执行状态，好做后期恢复。

- 问题解决

O2O 购物节期间发生的异常事件，会成立问题解决专项小组，严格跟进问题解决状态，直到问题解决才关闭。

## 战后

- 预案恢复

O2O 购物节结束后，针对 O2O 购物节期间执行过的预案进行恢复，并监控系统运行状态。

- 问题总结

O2O 购物节结束后，针对期间发生的问题进行总结和反思，制定方案解决问题。

- 性能优化

O2O 购物节结束后，针对期间发生的性能问题进行系统重构和性能优化。

## 作者介绍

何翔，2010 年加入苏宁，主要从事 JAVA 领域开发与设计工作，现担任苏宁云商 IT 总部多终端开发技术负责人，全面负责苏宁 API 网关管理与设计工作。从零开始搭建苏宁 API 网关，通过一次次系统重构与性能优化，经历了多次 818 促销和双十一 O2O 购物节的考验，支撑了亿级流量调用，保证系统稳定高性能。在 NIO、性能优化、高可用、扩展性、高稳定性等方面有一定的思考和领悟，目标是用技术打造一个高性能高可用的服务平台。

# 当当 11·11：高可用移动入口与搜索新架构实践

作者 王凯 等



每年一次的互联网电商的双十一，是对一年工作的总结和洗礼，在 2017 年的双十一到来之际，我们希望通过本文梳理一下当当的一些行之有效的方法。

面对爆发性增长的流量，首要任务是让系统在大流量冲击的情况下能够先存活下来，通过应用限流，您可以了解到当当如何在流量洪峰中保持现有系统的服务能力。

在众多的系统中，系统之间的交互过于繁杂，一套全方位的链路监控系统，可以快速定位某个子系统的问题，通过 APM 系统，您将了解到当

当在大量服务并存的情况下如何快速定位系统问题。

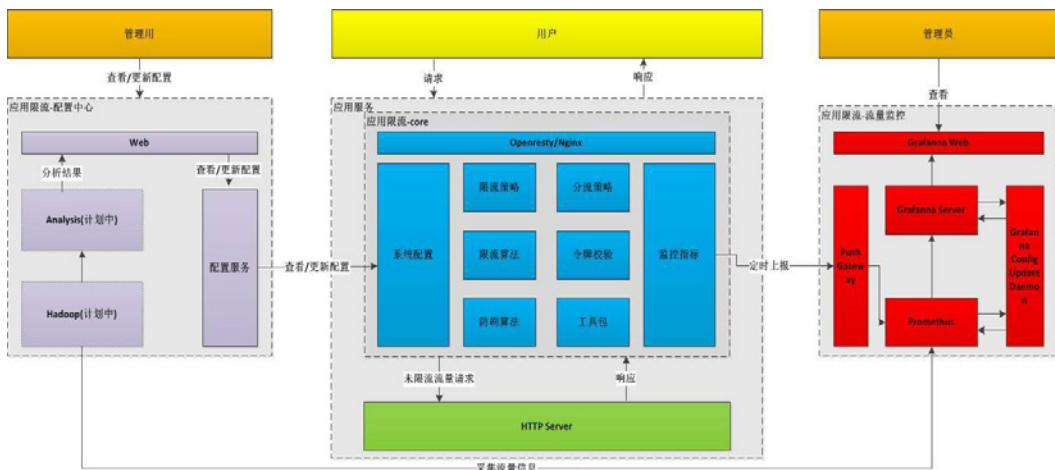
MAPI 是载当当大部分入口流量的移动端网关，通过 MAPI 对服务治理以及性能的分享，您将能够以点带面地一窥当当业务系统的设计思路。

搜索系统的重要程度不言而喻，而且它与业务系统的侧重点不尽相同，今年当当重点对搜索系统进行了重构，您可以通过对搜索新架构的解读一同探索技术的点点滴滴。

## 应用限流

电商平台的流量因促销、抢券、秒杀、热销品开卖等情况会出现大大小小的流量洪峰。流量洪峰的瞬时冲击（比如双十一零点的电商大促），可能导致系统响应缓慢，甚至整体崩溃，将对公司带来直接损失，因此大部分互联网电商都会都有自己的一套解决方案，比如：流量疏导、服务降级、应用限流等。接下来介绍下当当在应用限流的设计思路和一些做法。

限流体系的整体架构图如下：



当当的应用限流包含 3 个子系统：

- 基于 Openresty/Nginx 的限流引擎，包含限流策略和限流算法。
- 基于 Prometheus 和 Grafana 的流量监控模块
- 基于 Hadoop（计划中）和 java Web 的配置中心

下面分别介绍各子系统的一些设计经验。

## 限流引擎子系统

在实际业务中，绝大部分应用服务器部署在反向代理服务器（Nginx）后面。一般情况下，Nginx 的性能不会成为瓶颈，后端的应用服务器会由于各种原因（如数据库缓慢、I/O 等待时间长、代码自身的缺陷等）成为瓶颈，所以主要是根据 Nginx 后端连接的应用服务器的负载情况来决定采用何种策略对前面的 Nginx 进行限流。同时对应用服务器层无需做任何调整即可实现流量控制，开发简单，维护成本低。

反向代理服务器使用的是 Openresty (<http://openresty.org>)，一个基于 Nginx 的 Lua 环境封装，使用 LuaJIT 作为 Lua 编译器，由于 Just-In-Time 的方式，比传统 Lua 运行更高效。当的限流逻辑基于它开发。

首先介绍一下 限流算法，常用的拥塞控制算法有两种：

算法	描述
漏桶算法	用于控制从本节点发出请求的速率。
令牌桶算法	用于控制从外界发送到本节点的请求的速率。

漏桶算法可以用在控制应用服务器发送请求到其他应用服务器的速率，确保一个稳定的速率，主要用于控制回调洪峰（其他系统的调用）的，针对用户洪峰更适合令牌桶算法。

但是如果无差别拒绝请求，会导致用户体验的严重下降，因此需要策略组的配合，按照一定的规则拒绝请求，比较合理的做法是：

- 通过防刷算法，拒绝疑似攻击的请求

通过用户识别码、IP、所在区域、上网特征和接口特征等方式拒绝频繁访问的用户，保证服务器资源能服务更多的用户。

- 尽量不影响已服务用户的使用体验

对于正在挑选商品的用户（较多的单品页、添加购物车等操作），要尽量保证不被限流挡住，否则会降低用户的使用粘性。可以做的方法是增

加一个访问令牌，记录用户一些重要操作时间，在流量不够的情况下，优先放行具备访问令牌的用户。

- 尽量保证会员的使用体验

平台的会员尤其是高级会员一般是具备较大网购需求的用户，他们往往更能影响平台的口碑，所以平台要服务好这部分的会员。

其次将限流引擎中的参数进行整理，对外提供 HTTP Rest 接口，可以热更新限流策略和相关参数，对接应用限流 - 配置中心子系统。值得注意的是，openresty 提供了一种 dict 的类型是子进程内存共享的，可以跨 worker 访问，如果操作是进程不安全操作记得加锁，另外这种类型在声明时需要提供开辟内存大小，根据业务负载情况进行指定，超过内存大小会通过 LRU 算法进行内存回收。以下介绍了一些重要参数的说明：

参数	描述
限流开关	开关整个限流引擎，包含数据停止上报功能。
受限阈值	每秒低于此阈值可以任意访问，高于后将优先服务特点流量
拒绝阈值	超过后全部拒绝，保护服务器
突发阈值	允许一定量的突发请求，比如秒杀，促销开始时刻等
访问令牌	配置秘钥和失效时间
重定向	对拒绝请求定向地址
分流代理地址	符合分流策略逻辑的流量代理地址
上报数据	配置上报数据间隔和地址

最后提供了 流量指标上报 模块，按照配置参数进行定时数据上报，目前上报总流量、通过流量、拒绝流量、分流流量、主机 IP、应用名等实时信息。上报数据进入流量监控子系统，进行实时流量监控和离线分析，稍后会在流量监控模块详细说明。

## 流量监控子系统

Prometheus 作为一种 time series 数据库，很适合做实时业务监控。因此根据上文上报数据的配置，经过 Prometheus Push Gateway 将数据初步汇总，最终通过 Prometheus 拉取数据。

Grafana 则接入 Prometheus 数据源，提供实时流量页面和告警功能。可以通过查询各个应用集群的流量分布，通过率低于阈值或者一定时间流量为 0 则直接告警。

Grafana Config Update Daemon 则定时比对 Prometheus 和 Grafana 的配置差异，动态调整 Grafana 配置图，能够达到上下线主机和应用时，不需要手工维护 Grafana Web，实现自动化运维。

## 配置中心子系统

通过 web 页面调用推荐引擎子系统提供的接口，获取当前各个系统配置信息。并通过页面进行限流策略、限流阈值等信息的实时更新。未来计划通过定时抓取 Prometheus 的数据，将数据存放在 Hadoop 中，基于 time series 预测算法训练模型数据，对未来大促进行流量预判。通过提供的 Web 页面，管理员可以查看历史流量和流量预测数据，指导系统管理员优化应用限流配置的同时，也积累了宝贵的大促流量数据。为日后分析用户行为，甚至市场运营的判断提供更广阔的空间。

## APM 系统

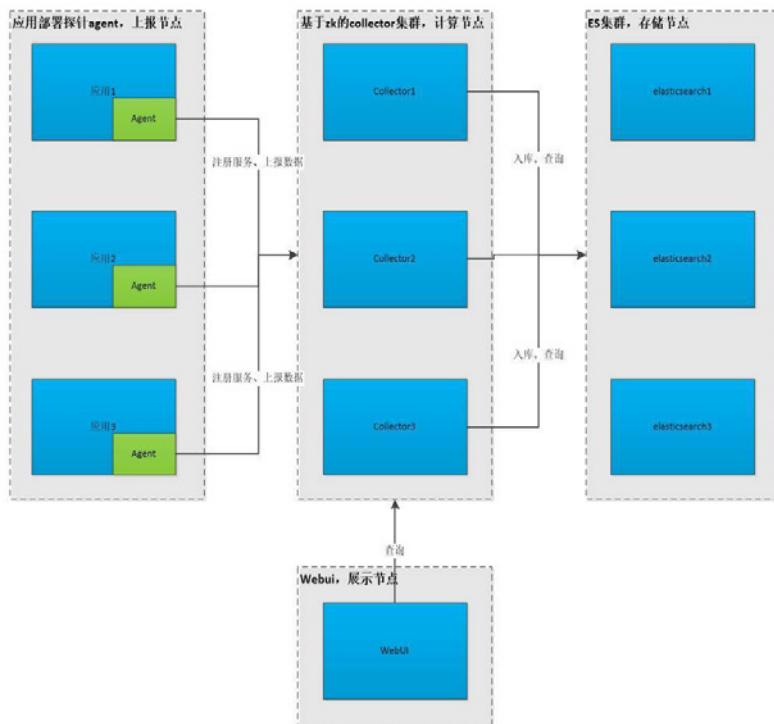
如今的电商平台随着业务复杂化，包含的功能越来越多，内部系统间的调用关系也越来越多。针对一个用户的一次购买流程（商品展示、搜索、购买、支付）可能需要调用数十次、甚至上百次的接口。首先如此庞大的调用链梳理成为一件很棘手的事情，而更迫在眉睫的事情是针对用户偶现的慢操作问题，无法快速定位到众多调用中的哪一步导致的。因此急需一套完整的应用性能治理系统——APM（Application Performance Management）来解决这个问题。

在 APM 产品选型时，我们考虑到当当目前系统的现状，得出了以下

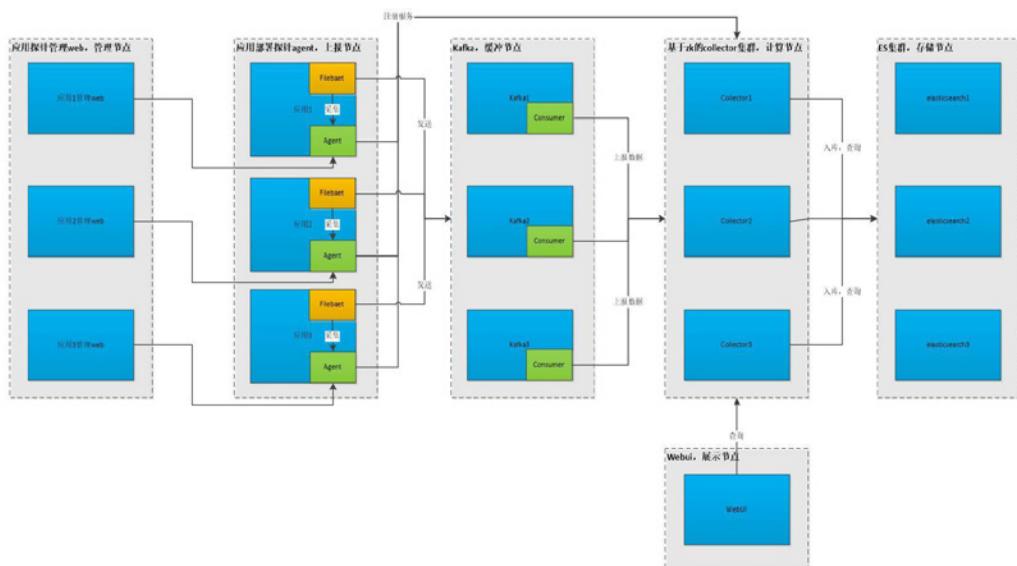
的结论：

- APM 产品需要性能极高，嵌入线上系统的探针需要尽量少的资源来完成调用信息获取，APM 自身也需要能承载百亿量级的入库和查询操作的能力。
- APM 产品需要方便部署使用，尽量不改动或者轻微改动代码即可实现系统追踪。
- APM 系统可以提供系统依赖关系、SLA 指标和调用耗时甘特图。
- APM 产品方便定制开发，方便扩展内部不同语言的系统使用和提供各种的统计数据。

最终我们选取了 OpenSkywalking 团队开发的 skywalking 产品 (<http://skywalking.io>) 作为当当 APM 系统，并进行定制开发。Skywalking 是遵守 opentracing 规范 (CNCF 基金会下，开源分布式服务追踪标准) 开发的，针对 java 支持使用字节码增强技术，无需改动代码即可实现系统追踪。自动追踪支持主流的 web 容器、中间件、RPC 组件、数据库和 NOSQL 等。架构图如下：



Skywalking 每个节点都支持集群模式，可以无限水平扩展，易用的 Java 探针自动上报数据，无需程序做任何改动。但是当当内部系统使用语言众多，除了 Java，包含 PHP、C、Python 和 Go 等语言开发的系统。对于 APM 系统来讲，追踪的调用轨迹只要一个节点没有追踪到，就会导致链路中断，达不到预期效果，因此我们在 Skywalking 上针对不同语言开发了相应的组件包 agent。agent 中为了尽量减少对业务系统的影响，使用写文件的方式，再通过 Filebeat+Kafka 的方式将追踪信息发给 collector，改造完的架构图如下：



如上图所示，agent 只负责汇总收集数据，和 collector 保持启动时候的注册功能，数据上报功能则通过写文件的方法，交给另一个 Filebeat 进程来完成。Filebeat 通过配置一个 Kafka 的 Topic，将数据发送到 Kafka 集群，再通过 Kafka consumer 来拉取数据发送 skywalking collector。同时 agent 提供管理服务 API，可以通过管理 web 界面管理 agent 的运行情况、开关、采样率等信息，在 agent 工作影响线上服务性能时（比如大促之时），可以快速关闭探针采集和上报功能来保证系统最大性能；各个系统也可以根据自身需要调节系统的采样率，达到追踪和性能影响的一个平衡点。改造后的优缺点如下：

- 优点

- a) 异步 Filebeat 进程发送数据，减少对业务进程的影响
- b) 更好的支持异构语言系统
- c) 探针支持热变更，更适合线上业务系统
- 缺点
  - a) 部署成本和复杂度提高
  - b) 数据计算会有更大延迟

目前当当 APM 系统已经上线，在 MAPI、购物车、交易、促销、TMS、搜索、价格、广告等系统上嵌入，进行 7\*24 小时全天候监控。在双十一前的几轮压测中，通过 web 界面提供了整个系统的 SLA 指标情况，对整体压测是否通过，提供了简单直观的数据。再通过查看慢操作的甘特图，可以快速定位具体的是哪个调用步骤比较慢，指导业务系统开发者快速发现问题所在，解决可能存在的性能瓶颈问题，也给整个电商平台性能更好的指导意见。

## MAPI 服务

MAPI 是当当移动客户端所有请求的流量总入口。MAPI 需要对客户端做安全、登录等移动端校验后，才可以将客户端真实的业务请求发送相应的后端服务进一步处理；MAPI 同时负责通过缓存机制减轻后端压力；MAPI 的另一个职责是管理移动客户端的用户信息，如：Session、设备、账号等，以及移动端的定制化服务，如：移动 APP 推送消息等。

### 微服务统一 API 网关

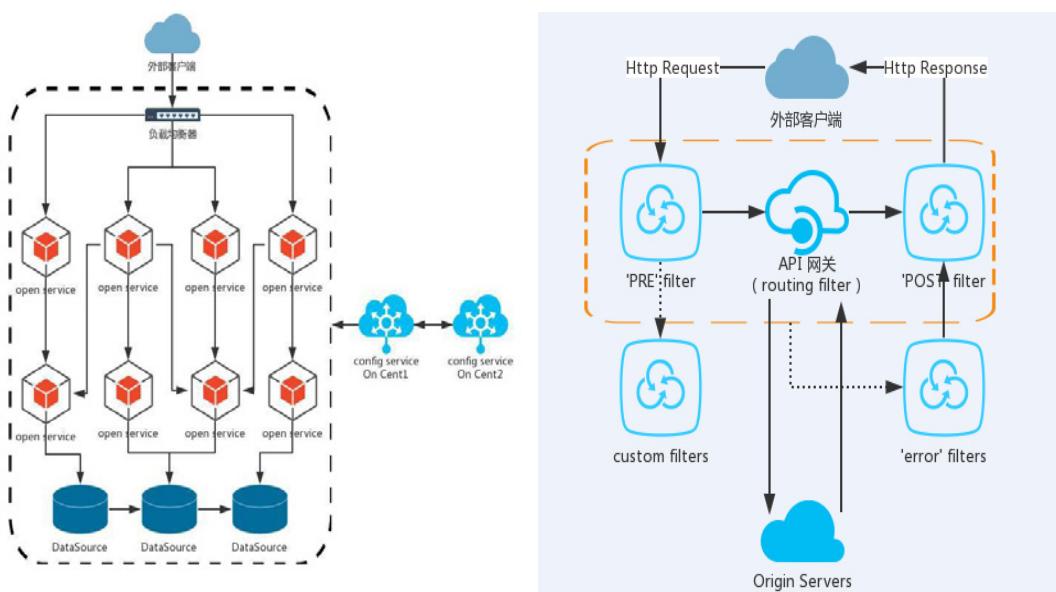
为了保证对外服务的安全性，我们在服务端实现的大量服务接口均加入了权限校验机制，并且为了防止客户端在发起请求时被篡改，引入了签名校验机制。我们遵循微服务架构设计理念，将原本处于同一应用中的多个模块拆分为多个独立应用。为保证不同应用提供的接口均需保证相同的校验逻辑，我们不得不在每个应用中都实现同一套校验逻辑。

随着微服务规模的扩大，校验逻辑愈加冗余，对它们的 BUG 修复或

扩展优化则必须在每个应用中分别修改。这不仅会引起开发人员的抱怨，更会加重测试人员的负担。所以，我们需要一套机制解决此类问题。

为此，我们采用了 API 网关架构的解决方案。API 网关定位为设计模式中的 Facade 模式，它作为整个微服务架构系统的门面，对所有的外部客户端访问进行调度和过滤。它不仅具有请求路由、负载均衡、校验过滤等功能，还有服务发现、熔断、服务聚合等能力。

系统架构改造前（左图）系统改造后（右图）：



## 服务治理

### 服务降级

在大促期间，为了减少服务器压力，会对一些相对消耗服务器计算资源，但并非核心流程的服务进行降级。客户端请求时，会请求降级服务，根据降级的配置来判断是否需要做相应的请求，如：购物车数量、单品缓存时间、请求收藏状态等。

### 熔断

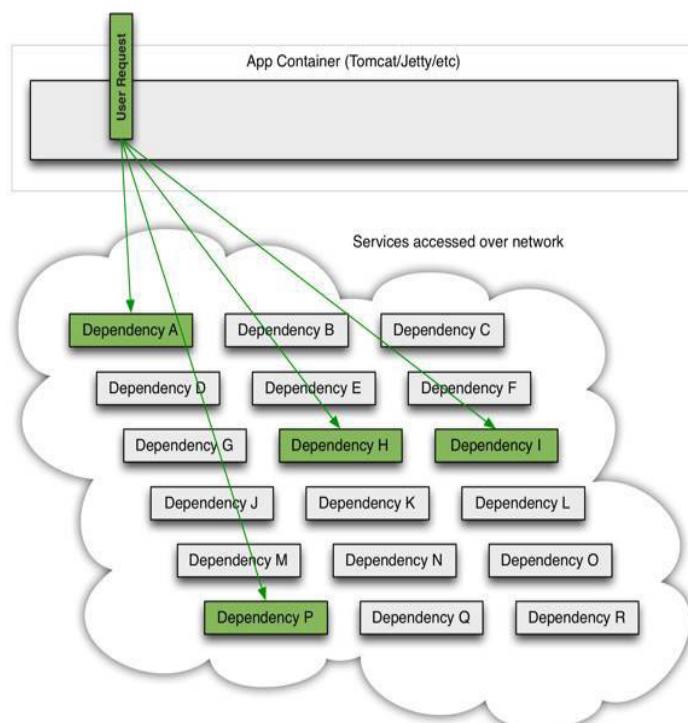
在微服务架构中，我们将系统拆分成了很多服务单元，单元之间通过服务注册与订阅的方式互相依赖。依赖间通过远程调用（如：RPC/

Restful API 等）的方式交互，这样就有可能因为网络原因或者依赖服务自身问题出现调用延迟或异常，最后就会因等待出现的故障的依赖方响应形成请求积压，最终导致服务自身的瘫痪。

以互联网电商为例，系统会拆分为用户、订单、库存、积分、评论等一系列服务。在用户下单时，客户端将通过 MAPI 接口的调用，请求订单服务的创建订单接口，订单服务则会请求库存接口。如果库存或订单服务因网络延迟等原因造成响应超时，MAPI 接口的线程将被挂起。高并发场景中会造成挂起线程的大量堆积，导致客户经过漫长等待而下单失败。过多的客户下单请求将产生雪崩效应，导致服务不可用，甚至整个系统瘫痪。

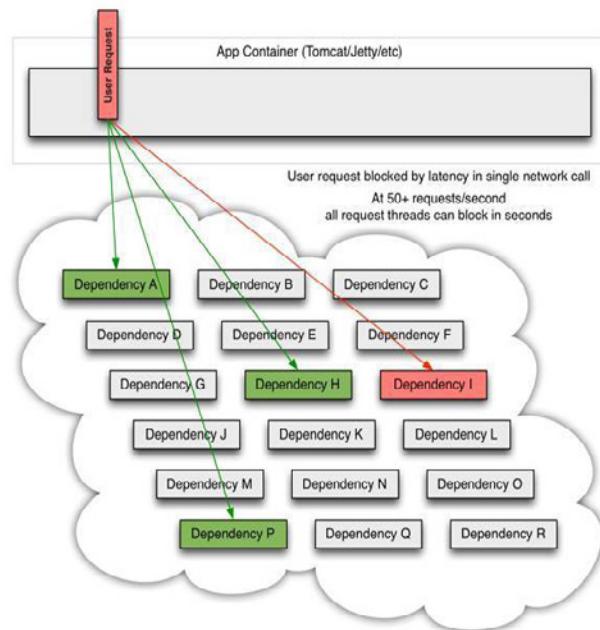
当当采用 Netflix 开源的 Hystrix 作为熔断方案，以下对 Hystrix 进行分析讲解。

下图展示了用户请求正常访问下的一个情况：



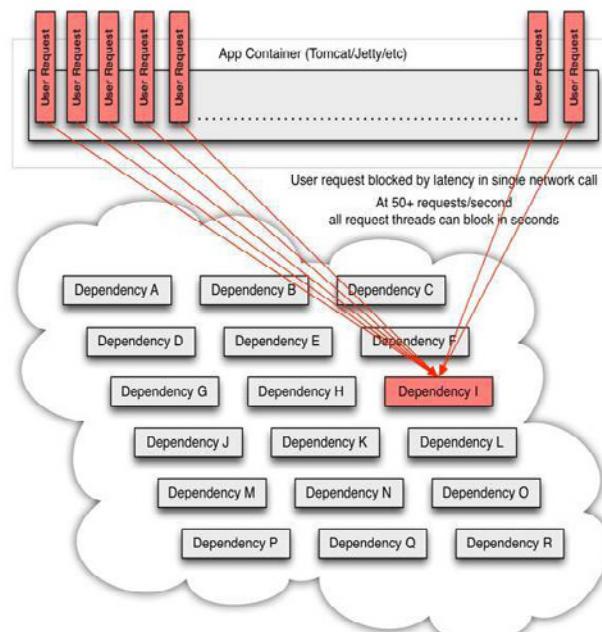
（图片来源于 GitHub-Hystrix）

当其中的某一个系统发生延迟，并阻塞用户请求时，如下图：



(图片来源于 GitHub-Hystrix)

在高并发的请求的情况下，会造成所有的用户请求发生延迟，最终整个消费系统瘫痪而不可用，如下图：



(图片来源于 GitHub-Hystrix)

Hystrix 使用“舱壁模式”实现线程池的隔离，它会为每一个依赖服务创建一个独立的线程池，这样就算某个依赖服务出现延迟过高的情况，也只是对该依赖服务的调用产生影响，而不会拖慢其他的依赖服务。

## HHVM 提升服务性能

为了满足快速的迭代开发，MAPI 中大部分采用 PHP5.6 来实现的，其中有一些使用了 Hack 的 IO 异步操作。我们经过大量的压测案例对比，发现 HHVM 更加符合 MAPI 业务，故我们采用 HHVM 虚拟机。

同时，HHVM 的作者（Facebook 的同事）主动与当当进行技术交流，更有效地提升了 HHVM 应用层性能，使得业务高效运行。

HHVM 类似于 C# 的 CLR 和 Java 的 JVM。HHVM 是在 HPHPc 的基础上构建，它会将 PHP 代码转换成高级别的字节码（一种中间语言），在运行时即时（JIT）编译器会将这些字节码翻译成机器码，它比 open\_cache（Zend 使用的）更稳定，更高效。

## 搜索新架构

今年双十一，当当网上线了搜索新架构。平均响应时间降低了一个数量级，QPS 提升了一个数量级，索引数据的实效性也有了大幅度的改善。可以更加从容的应对双十一的流量，提升用户的搜索体验。

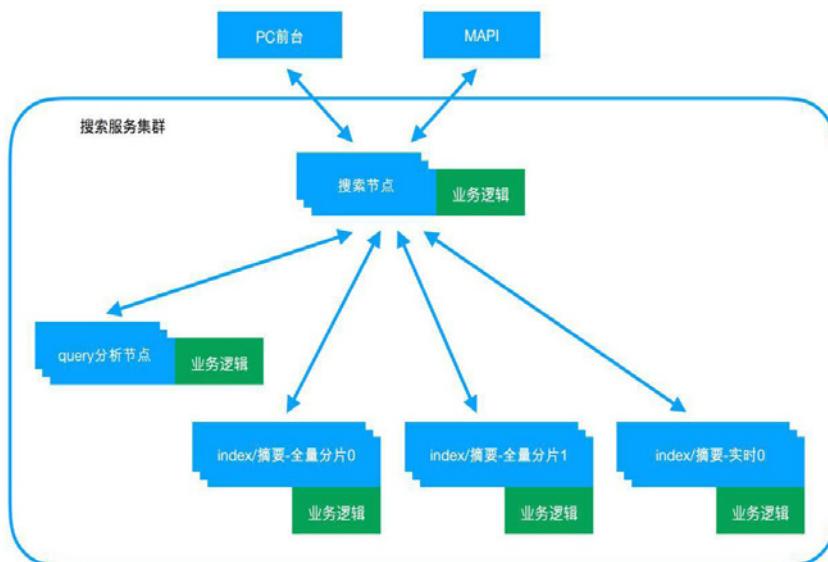
当当网的搜索引擎是典型的电商搜索引擎，索引量远小于互联网搜索引擎，不会达到数百亿，但有很多复杂的电商搜索逻辑，可以说是术业有专攻。搜索引擎全部由当当自主研发，主要使用 C++ 语言，长期以来，支撑当当搜索业务的同时，也积累了一些技术债。为了快速响应业务需求，导致业务逻辑的无序添加，由于缺乏有效的定期梳理，积重难返，搜索效果迭代的效率越来越低。

16 年底，技术部启动了搜索引擎的重构项目。重构的技术路线面临多种选择，在人力有限的情况下，是全面转向开源技术，还是自主研发，存在争议。在充分调研不同技术路线的风险和收益后，搜索引擎的不同子系

统，采用了不同的方式：搜索服务对性能和稳定性有着极高的要求，现有开源框架不足以支撑，并且业务的定制能力也是一个问题，因此依然采用了全部自主研发的方式；离线系统对性能没有过高的要求，全面转向了开源技术，可以更加方便地实现业务逻辑、加快效果迭代的速度。

## 搜索服务重构

搜索服务在重构之前是一个单机版的检索程序，平均响应时间、索引量都存在极限，无法水平扩展。重构后，采用了分布式架构，拆分了搜索节点、query 分析节点、index 节点、摘要节点。各节点之间，没有依赖关系的运算可以并行，大幅度降低了平均响应时间；index 节点拆分后，索引数据可以分层分片，索引量不再受单机限制。



搜索新架构没有沿用旧系统的设计和代码，重新设计编写了从底层倒排索引、索引合并、属性筛选和汇总，到最终结果合并、排序的全部代码。通过代码的重新编写全面梳理了业务逻辑，严格划分了搜索框架和业务逻辑的代码边界。实现了一个通用的搜索框架，在这个框架之上，附加了当当网的全部搜索业务逻辑。搜索框架和业务逻辑的剥离，为效果迭代打下了坚实的基础。

搜索服务重构耗费了很多时间，最终取得了明显的收益，单机检索性能有了大幅度的提升：平均响应时间降低了一个数量级、QPS 提升了一个数量级。分布式架构是一个因素，索引结构、属性筛选和汇总、排序方式的改变也是另一个重要因素。

## 索引实时更新

电商搜索对于实时性有着极高的要求，尤其是双十一，价格、库存的变化需要及时更新到索引中，商品的上下架、标题等促销信息的修改也需要快速生效，搜索新架构充分考虑了这些应用场景，做了专门的优化。

搜索旧系统在索引数据的更新上有 2 个问题：一个是更新需要加线程锁，更新的瞬间全部查询线程暂停；另一个是采用传统的全量 + 增量索引，增量会显著影响 term 的 doc 链合并速度，导致索引结构劣化，一个全量周期内，增量商品的数量不能太多，否则就会导致检索耗时过长。

搜索新架构采用了新设计的索引结构，更新不需要加线程锁，可以在并发查询的同时更新索引数据，大幅度提高了更新效率；同时采用全量 + 实时索引，替换了全量 + 增量索引，独立的实时索引节点使得索引结构不会劣化，增量商品的数量不会显著影响检索耗时。

新架构的实时索引常驻内存，由于需要动态增加商品，整体上是链式的，采用跳表 + 数组加速。这里有一个平衡，全链式存储效率高，没有内存空间浪费，缺点也很明显，由于它是跳跃式的，CPU 的 cache 命中率极低，term 的 doc 链合并过程耗时过长；跳表 + 数组虽然耗费内存空间，但会明显提升检索效率和 CPU 的 cache 命中率，缩短检索耗时。

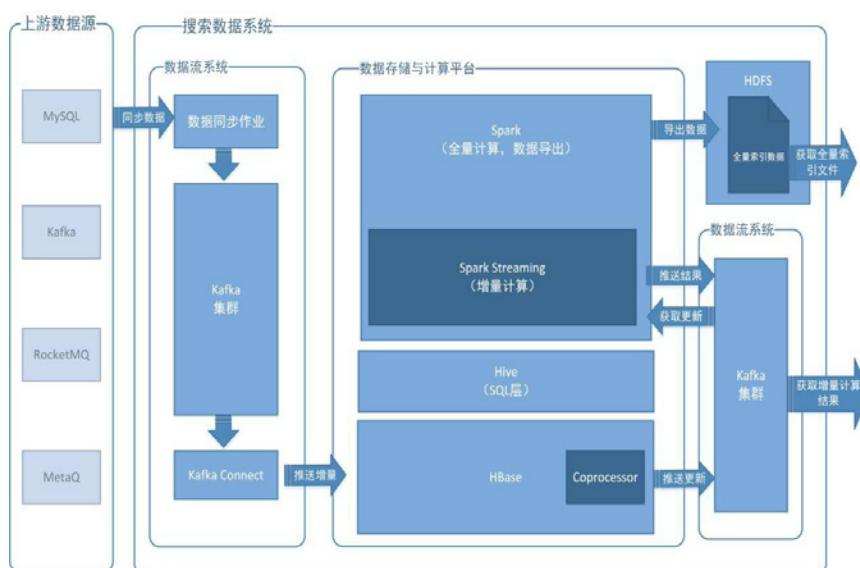
搜索新架构的实时索引通过参数调整内存耗用和检索耗时，在索引量相同的情况下，对比全量索引，能够控制在 2 倍的内存空间耗用和 4 倍的检索耗时。由于实时索引的能效比没有显著下降，仍在可接受的范围内，对于时效等级特别高的商品，可以去掉全量，全部用实时索引替代，简化建库、更新流程。

## 离线系统重构

搜索引擎的离线系统，负责收集搜索服务需要的全部数据，当当网搜索引擎的离线系统，需要汇集商品的全部信息：标题、描述、价格、库存、销量、分类、属性、评论、店铺、促销、各种需要展示的辅助信息等。这些信息来自多个上游系统，接口方式异构，有数据库表的，也有消息队列的，还有文件形式的，各自有着独立的更新周期，之间还存在依赖、绑定关系，业务逻辑十分复杂。

搜索旧系统采用 MySQL 作为主数据库，依赖多种语言编写的同步作业对接上游系统，绝大部分数据最终汇合到主数据库。全量索引相对简单一些，直接从 MySQL 拉取全部数据；增量则麻烦一些，需要依赖一个自主研发的数据中心，通过比对检测价格、库存、标题等实时数据的变化，推送到搜索服务。

重构前，MySQL 主数据库和数据中心是整个离线系统的瓶颈：MySQL 作为主数据库，优点是关系数据库添加业务逻辑方便，但缺点也很明显，即不适合搜索场景，主要是全量数据的 dump 耗时过长；数据中心是单机版程序，受内存容量限制，能够检测的索引量存在上限，本身结构复杂，添加业务逻辑的成本过高。



重构后，整个离线系统全面转向了分布式，用 HBase+Spark+Kafka

的大数据开源框架替换了 MySQL 主数据库和数据中心：主数据库采用 HBase 集群，明显降低了全量数据的 dump 耗时；同步作业只负责同步数据，最大限度地剥离了业务逻辑；业务逻辑全部采用 Spark 重新编写，研发效率高；增量采用 HBase 的 Coprocessor 检测数据更新，用 Kafka 做数据流转。重构后的离线系统，运行效率明显提升，全量建库时间缩短到旧系统的 1/4，绝大部分索引字段的更新缩短到了分钟级，大幅度改善了索引数据的实效性。

## 小结

通过以上几个系统，我们复盘了当当一年来的进展以及对双十一的准备。篇幅有限，有不少新的变化无法一一说明，目前当当基于 Mesos 的作业云平台，已经承载了上万的业务作业；容器编排也进入了最后的落地阶段；Service Mesh 的探索正在进行中；期待下一次的双十一可以分享更多的实践经验。

# 有赞 11·11：全链路压测方案设计与实施详解

作者 金瑞敏



每年双十一，对于买家来说是一年一度的购物狂欢，可是对于电商公司的技术人员来说，却是一年一次的大考。如何用更少的预算完成指定当前业务规模的流量高峰，是技术的永恒主题。

有赞在双十一之前完成了全链路压测方案，并把它用于大促的扩容和容量验证，取得了不错的成果。

在电商公司待过的技术同学都知道，在大促来临时，整个集群的最高峰压力将是正常时间的几十倍，最高峰持续的时间会特别短，然后回落到正常水平的几倍。所以，我们可能会自然而然地想到，把整个集群扩容几

十倍的机器，在双十一当天应对几十倍的流量，然后第二天减至正常量，就可以完成大促的考验。事实情况是否真的这么简单？

## 大促保障的困难

用户购买商品的链路是一条很长很复杂的系统集群，中间会涉及到店铺、商品、会员、营销、交易、支付等 6 大核心模块，每个模块又会涉及到多个不同的服务化系统单元，我们把这一条骨干的链路就叫做核心链路。

大家都知道，双十一当天，真正爆增的其实是买家的购买量，像开店 / 商品上架等功能，其实并发量没什么变化。也就是说，真正的压力其实是在核心链路上面，如果把所有的系统都扩容几十倍，本身就是一个很大的浪费。正常来说，一个稍有规模的电商公司，日常有几千台机器维持正常的运转，本身就是一个较大的开销，如果突增几十倍的系统开销，对于公司的财务也是很大的压力。所以，一个较理想的方法，是只把核心链路的系统扩大几十倍的系统吞吐量，就可以达到目标。

### 困难一

大型的分布式系统其实错综复杂，公司需要维持成百上千的服务化系统。理论上来说，只有少部分系统是核心链路的系统。但是在实际情况下，因为公司人员的关系，可能会把某些非核心系统，不知不觉加入到了核心链路中。所以，第一件要做的事情，就是把非核心系统从核心链路上剔除。

### 困难二

一般公司都会在线下搭建性能测试环境，在该环境下，我们的测试同学可以借助一些测试工具，去压单机单接口的性能。假如，店铺的首页面，我们在性能测试环境下，得出单机单接口的 QPS 峰值是 500，这是否意味着，要达到 10w 的 QPS，我只需要设置 200 台机器就可以了呢？答案当然是否定的。因为任何的页面接口都不是单独存在的，都会受到公共

资源的制约，如：DB、Redis、MQ、网络带宽等。比如当店铺首页达到 10w QPS 的时候，商品详情页可能要达到 8w 的 QPS 了，也会消耗公共资源。当公共资源的能力下降时，势必会影响所有的系统。所以，通过单机性能算出来的理论值，跟实际情况会相差很远。

### 困难三

正常来说，任何大促都会有业务目标，这个目标一般是拿 GMV 进行评估。但是我们在进行系统容量评估的时候，一般会想扩大多少台机器。那么 GMV 跟核心链路各个系统之间的机器数量的转化关系是什么样的？

### 困难四

做过大型分布式系统的同学，可能都知道一个事实，即整个集群的性能其实取决于接口的短板效应。而这个短板的接口，在正常的流量下，是不会显现出来的。只有集群的整体压力达到一定值的情况下，才会偶尔显现，然后造成雪崩效应，拖累整个系统集群。所以，如何在大促之前找到这些短板，然后把它们一个一个优化，这件事情就显得非常重要。

### 困难五

应用系统的扩容相对而言是比较简单的，完成大促之后，可以很容易归还。但是 DB 等核心资源的扩容其实并不容易，而且资源不可能归还（数据不可丢失）。

事实是检验真理的唯一标准，上面提到的五个困难，其实都可以用线上真实压测的办法去检验。业内大型电商公司，会用全链路压测的方案去指导扩容的进程，有赞也不例外。今年双十一，有赞用该方案完成了对核心链路 20 倍的扩容，但是整个集群的规模只是扩大了一倍多一点。

## 有赞全链路压测的设计方案

全链路压测的目标是让双十一要发生的事情提前发生，并验证在该情况下系统表现良好。做线上压测，有一个很重要的原则：线上系统是不允

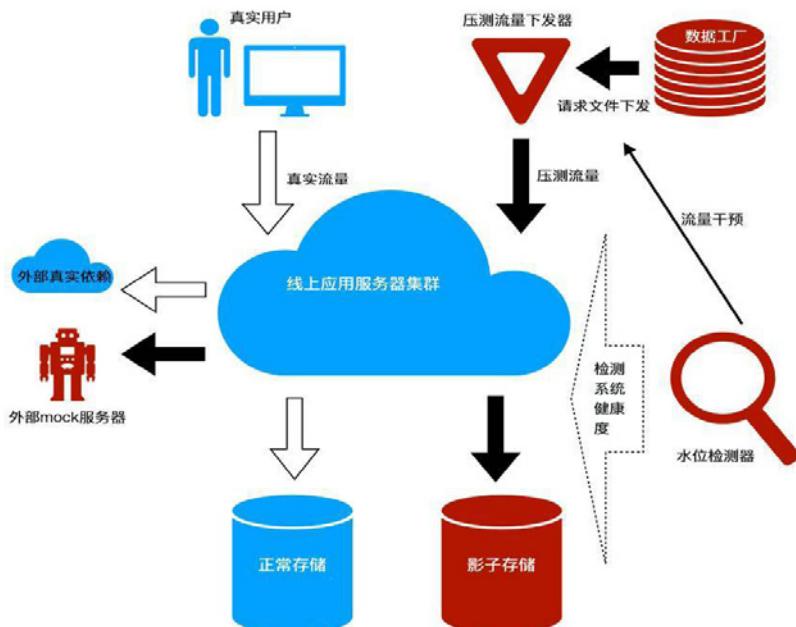
许有脏数据的。

有赞的压测设计方案，可以用几句简单的话做概括：

- 压测流量落影子库，正常流量落正常库。
- 压测不能把系统压崩溃。
- 压测是模拟双十一流量最高峰时用户的购物行为，所以综合性的流量模型需要跟实际情况相符。

## 全链路压测的总体设计

有图有真相，我们先上图。



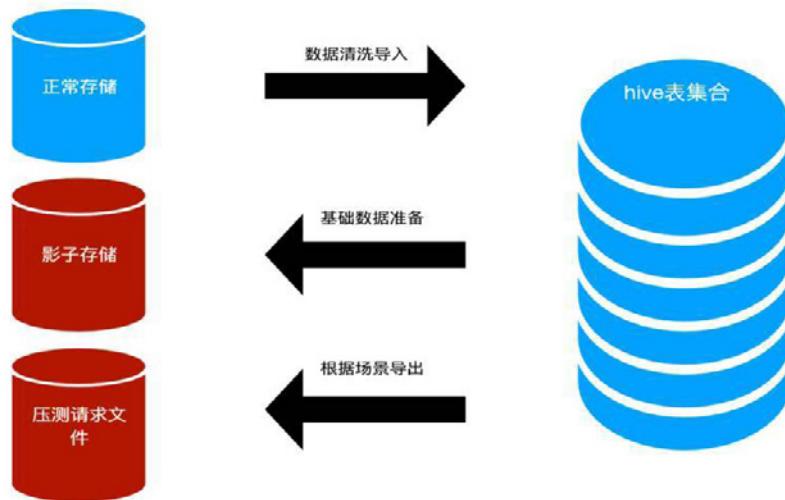
在上述图中，我们明显可以看到，全链路压测有几个关键部分：

1. 数据工厂，负责造请求数据。
2. 大流量下发器，产生很大的压力去压系统。
3. 线上服务集群同时处理压测请求 + 正常请求。
4. 压测流量落影子存储，正常流量落正常存储。
5. 压测流量对于外部的依赖走 mock 服务器，正常流量走正常外部集群。

- 水位检测，需要检测存储 + 线上应用服务器的健康度，并且能够干预流量下发。

## 关键模块的设计方案

### 数据工厂设计



数据工厂是压测的一个核心部件，主要由 Hive 表的集合 + 各种导入、导出脚本组成。

数据工厂的目的是保存压测需要准备的所有数据，数据需要做清洗，比如：

- 商品未下架
- 商品的库存无限
- 营销活动的信息有效，未过期
- 店铺未关闭等等

场景的定义：场景的定义关系到数据的准备，正常来说，压测只会压随着买家人数暴增、系统的压力立即增加的场景，我们把这个场景涉及到的系统叫做“核心链路”。

### 影子存储的设计与路由能力

- DB、路由方式由 RDS 提供，存储可以有两种方式：

1. 影子表与正常表存在同样的 instance、不同的 schema 中。这个方案不需要增加额外的存储开销，相对更便宜，但是风险较高（把库压死了会影响线上业务）。
  2. 影子表与正常表存在不同的 instance、相同的 schema 中。这个方案相对较贵，需要额外搭建 DB 集群，但是安全性较高。我们选择的是第一个方案。
- Redis：通过 key 值来区分。压测流量的 key 值加统一前缀，通过 Redis-Client 做路由。
  - HBase：通过命名空间做隔离。影子空间加前缀，提供统一的 HBase Client 做数据访问，由该 Client 做路由。
  - ES：通过 index 名字来区分。影子的索引统一加前缀，提供统一的 ES Client 做数据访问，由该 Client 做路由。

### 线上应用集群的变更

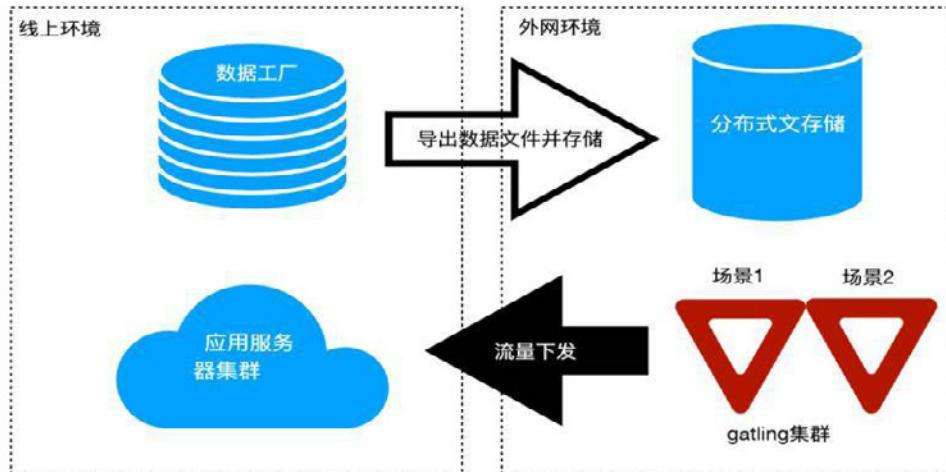
- 统一线上应用对于数据的访问（DB+ES+HBase+Redis），提供统一的 Client。
- 由于线上的应用都是服务化工程，远程调用时，必须具备压测流量的标记透传能力。
- 线上的少部分应用，需要访问第三方服务，比如：物流、支付。这些应用需要改造为压测的流量直接访问 mock 服务器。

### 全布式流量下发器设计与链路设计的能力

我们选用 gatling 作为我们的流量下发器实现。

### 数据文件的内容

每一种场景都有不同的数据文件，数据文件由场景相对应的多种 url 组合而成。比如：我们本次压测会压“无优惠的场景、秒杀场景、满减场景、拼团场景”等等。无优惠的场景分为“店铺首页、商品详情页、加购物车页、下单页、支付页、支付成功页”等等。这个文件不涉及漏斗转化率。一般来说，一个数据文件很大（至少是 G 级别的）。所以我们的数据文件内容格式为：



### 所有的数据文文件

- 场景集合 1
- 场景集合 2
- 店铺首页 URL
- 商品详情页 URL
- 加购物车页 URL
- 下单页 URL
- 支付页 URL
- 支付成功页 URL
- 店铺首页 URL
- 商品详情页 URL
- 加购物车页 URL
- 下单页 URL
- 支付页 URL
- 支付成功页 URL
- 无优惠的场景数据文件
- 秒杀场景数据文件
- 满减场景数据文件
- 拼团场景

## 流量下发脚本的内容

流量下发脚本的核心是控制漏斗转化率：

1. 不同场景的流量配比。
2. 每个场景下面，url 从上往下的漏斗转化率。

gatling 提供天然的转化率配置脚本，用起来非常方便。有兴趣的同学可以自行 Google。

## 水位检测系统的能力设计

这个是一个很重要的模块，在项目启动之初，我们希望以实时计算的方式，一边采集各个应用系统的资源使用情况 + 接口耗时 + 业务正确率，一边向 gatling 发送流量干预信号，以做到自动保护系统的目的。由于时间关系，我们并未实现这一方案。取而代之的是人肉查看实时监控界面的方式，人为去干预 gatling 的流量下发情况。

## 全链路路压测的实施

如果实施过全链路压测的项目，大家都会有一个共同的感受：做基础的组件容易，让核心业务去完成相关的升级与验证工作很难。原因只有一个：需要用全链路压测的公司，业务规模都很大，涉及的团队会特别多。梳理理清楚庞大的业务，让所有的业务团队一起发力，本身就是一件很难的事情。

我们把链路压测的实施分为以下几个阶段：

1. 基础中间件开发，各种框架升级开发，压测器研究与脚本开发；
2. 业务升级与线下验证（人工点击，数据落影子库）；
3. 业务升级与线上验证（人工点击，数据落影子库）；
4. 数据工厂数据准备；
5. 小流量下发验证（用 gatling 下发，数据落影子库）；
6. 大流量量压测与系统扩容。

第 2、3、5 阶段，需要借助业务测试同学的力量；第 4 阶段，需要借助业务开发同学的力量；第 6 阶段，则需要借助业务团队 + 运维同学的

力量。

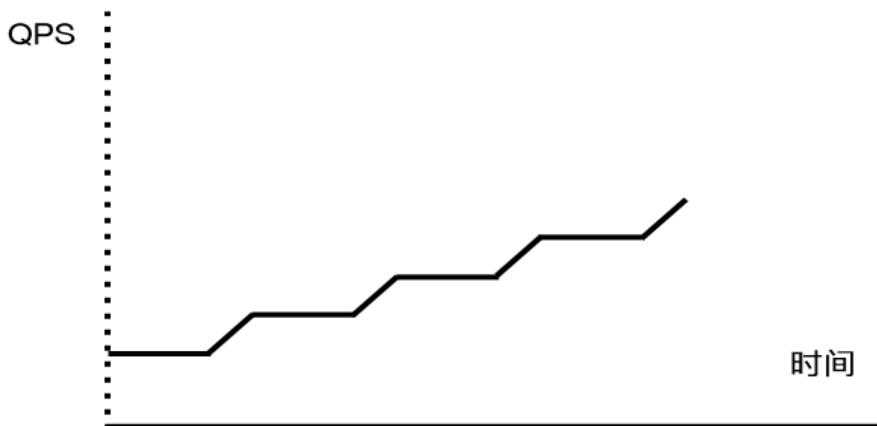
由于每个阶段人员都不太一样，所以需要每一个阶段都组织不同成员的虚拟小组，借助各个团队的力量完成相应的工作。

## 压测过程中的重要细节与把控

### 流量爬升的规律

正常来说，在大促之前做压测，目的的一般是给扩容 / 优化做方向性的指导。

假设我们双十一需要扩大 20 倍的容量以应对高峰，那我一定不会一开始 就拿 20 倍的流量去压我们的系统，因为这样做的话，所有的系统都会在一瞬间就挂掉，这样没有任何意义。我们的做法是，阶段性的爬坡打流量，然后把系统的能力一段一段提升上去。



例如：

1. 第一天，我们会以日常流量的最高峰为起始流量，然后爬坡到一个流量高峰 A，记为第一天的目标。在压测之前先做一次扩容。在压测中，碰到了某个瓶颈了，通过增加该系统的机器来提升能力。
2. 第二天，我们以 A 为起始流量，然后再次爬坡到 B。同样压测前做扩容 + 压测中碰到瓶颈加机器。

3. 以此类推，一直到最终流量达到目标流量为止。
4. 每一天的压测，也需要以慢慢爬坡的方式提升流量。在爬坡的某个高度稳定 5 分钟，然后再次爬坡。稳定时间 5 分钟，爬坡时间 30 秒。

### 非核心链路进核心链路的问题

发现并解决这个问题，本身就是压测的目的之一。

正常来说，非核心链路，在大促来临时不会扩大多少容量。当压测的压力增大时，很容易通过系统报警查到。

当发现这个问题的时候，一定要坚决要求业务方做系统改造，把非核心系统的强依赖去掉。解耦的技术有很多，需要根据不同的业务规则来选择方案。比如：业务降级、通过中间件解耦、异步化等。

### 上下游扩容 / 代码优化的选择

一般来说，在压测的过程中，当碰到压测流量不能再升高的时候，会有很多原因，我们碰到的情况有以下几种：

1. 下游的某些服务化工程的能力达到瓶颈了，导致网关 RT 值升高，拖累整个集群的 QPS 上不去。
2. 网关应用自身的能力达到瓶颈。
3. 中间件 /DB 能力达到瓶颈。
4. job 的能力达到瓶颈，导致数据处理不够及时。
5. 流量集中的页面，消耗了集群大量资源，如：店铺首页、商品详情页等。

针对 2、3、4 这样的情况，我们的选择是毫不犹豫地加机器，代码优化的性价比较低。

针对第 1 种情况，需要做一些分析，如果这样的能力是在系统设计者的预期之内的，可以选择加机器，如果完全超乎意料的，一定需要通过程序优化来提升能力，否则加了资源，可能还是瓶颈。

针对第 5 种情况，一定要做的事情是静态化。因为这些流量集中页面，一般都是展示性质的。不管如何做应用内的优化，获得的能力提升远

不如做静态化的收益大。

## 未来展望

全链路压测的方案有赞只是初试牛刀，我们已经看到了这个方案在提升 + 验证集群处理能力方面巨大的价值。当前，这个方案做得还较粗糙，存在一些问题：

1. 压测只能在夜间做。
2. 压测中需要有很多业务开发人员陪同。
3. 链路规划复杂度太高。
4. 压测控制台的稳定性还不够高。
5. 水位检测与流量干预是通过肉眼观察监控来实现。

后续我们团队会继续投入大量精力去完善整个方案。希望可以将压测方案变成：

1. 线上测试链路的机器人，实时检测线上系统的正确性，同时没有脏数据干扰。
2. 测试同学手里的工具，做到流量压测常规化，开发同学不用陪同。
3. 压测可以在白天进行，晚上熬夜毕竟不利于健康。
4. 链路规划图形化，并与数据工厂结合，完成数据的准备工作。
5. 通过水位检测与流量干预来保护系统，让业务系统不会被压崩溃。

## 作者介绍

**金瑞敏**，有赞核心交易、java 框架团队负责人。带领团队完成核心交易平台化体系建设，优化有赞服务化治理方案、环境隔离方案、全链路压测方案等等。长期从事分布式系统的建设与研究。

# 版权声明

**InfoQ** 中文站出品

架构师双十一特刊：电商大促技术探秘

©2017北京极客邦科技有限公司

本书版权为北京极客邦科技有限公司所有，未经出版者预先的书面许可，不得以任何方式复制或者抄袭本书的任何部分，本书任何部分不得用于再印刷，存储于可重复使用的系统，或者以任何方式进行电子、机械、复印和录制等形式传播。

本书提到的公司产品或者使用到的商标为产品公司所有。

如果读者要了解具体的商标和注册信息，应该联系相应的公司。

出版：北京极客邦科技有限公司

北京市朝阳区来广营容和路1号院1号楼叶青大厦北园5层

欢迎共同参与 InfoQ 中文站的内容建设工作，包括原创投稿和翻译，请联系 editors@cn.infoq.com。

网 址：[www.infoq.com.cn](http://www.infoq.com.cn)



极客邦科技



关注极客邦科技动态

## 极客邦科技介绍 >

极客邦科技（Geekbang）是一家世界级 IT 内容综合服务集团。创立于 2007 年，集媒体、会议、电商、培训、咨询、图书出版、社交、整合营销、创新孵化等九大产业于一体，总部设于北京。致力于让创新技术推动社会进步。十年内，极客邦科技为 100 万技术人，3000+ 中国企业提供服务，业务遍布中国各城市，以及美国、法国、德国、荷兰、以色列、日本等国家。

## 我们的产品和服务 >



### 极客时间 App

专栏订阅、实战课程、二叉树视频、极客新闻、图书、电商、垂直搜索等



### CTO 圈子

EGO 技术领导者社交平台、中国技术开放日全球行



### 培训 / 咨询

线下课程、企业内训、产品 / 技术咨询服务



### 技术会议

QCon 全球软件开发大会、ArchSummit 全球架构师峰会、AICon 全球人工智能技术大会、GMTC 全球移动技术大会、CNUTCon 全球运维技术大会、GTLC 全球技术领导力峰会



### 媒体资讯

InfoQ 网站、新媒体矩阵（InfoQ / AI 前线 / 聊聊架构 / 移动开发前线 / 细说云计算 / 前端之巅 / 高效开发运维）、二叉树原创视频、极客 Live 直播



### 整合营销

技术品牌打造、产品 / 技术推广、商机发现 / 挖掘、品牌形象包装、生态建设、新媒体营销解决方案、分享 / 活动 / 大赛、人才招聘、内容共建

## 联系方式 >

### 内容合作

content@geekbang.org

### 业务合作

hezuo@geekbang.org

### 反馈投诉

feedback@geekbang.org

### 咨询电话

010-64738140