解读 REVIEW

2015

Spark

政企私有云市
场风云录

大数据的黄金时代

Hadoop作为大数据平台中最基础与重要的系统，在2015年提高稳定性的同时，发布了多个重要功能与特性，这使得Hadoop朝着多类型存储介质和异构集群的方向迈进了一大步。

HDFS 之前是一个以磁盘单存储介质为主的分布式文件系统。但随着近几年新存储介质的兴起，支持多存储介质早就提上了日程。如今，HDFS 已经对多存储介质有了良好的支持，包括 Disk、Memory 和 SSD 等，对异构存储介质的支持，使得 HDFS 朝着异构混合存储方向发展。

YARN作为一个分布式数据操作系统，主要作用是资源管理和资源调度。在过去一年，YARN 新增了包括基于标签的调度、对长服务的支持、对 Docker 的支持等多项重大功能。**HBase**在 2015 年，HBase 迎来了一个里程碑——HBase 1.0 release，这也代表着 HBase 走向了稳定。HBase 新增特性包括：更加清晰的接口定义，多 Region 副本以支持高可用读，Family 粒度的 Flush以及RPC 读写队列分离等。

Spark: 2015年的Spark发展很快，JIRA数目和PR数目都突破了10000，contributors数目超过了1000，可以说是目前最火的开源大数据项目。

云计算

打磨产品服务
迎接市场变局

多深刻而又复杂的变化，2015”年终技术盘点系列 晰地梳理出技术领域在这

容器

扩张与进化

整个IT技术领域发生了许 InfoQ 策划了“解读 文章，希望能够给读者清 一年的发展变化，回顾过去，继续前行。回顾2015年容器技术的发展历程，我们可以用两个关键词来概括：**扩张与进化**如果说2014年仅仅是**Docker**为主的容器技术在**云计算**以及**DevOps**圈初露锋芒的话，那么2015年则是以Docker为核心的容器生态圈迅猛扩张的一年。

Golang 的全进发时代

并发的GC: 一般认为是garbage collector的缩写形式，通常被译为垃圾回收器。不过，它有时候也被看做是garbage collection的缩写形式，中文译为垃圾回收。在下文中，当GC被当做动词用时，指的是垃圾回收。但当GC被当做名词用时，指的是垃圾回收器。

InfoQueue

卷首语

郭蕾 Gary
InfoQ 主编

回顾这一年，互联网圈发生了很多有意思的事，国家把创业和创新放到了一个新的高度，在整体经济形式不好的情况下，互联网公司的发展却没有受到大的影响，创业公司如雨后春笋似地发展。与此同时，国内很多领域的巨头都开始合并，比如携程和去哪儿网，美团和大众点评，滴滴和快的，美丽说和蘑菇街，这也说明了另外一个现象，行业内的竞争非常残酷，大家也都开始求同存异，谋求共同发展。

创业热潮、抱团取暖，这是2015年整个互联网行业的主旋律。在这样的大背景下，InfoQ编辑分析了相关技术领域以及生态圈企业的发展，试图找到他们之间的联系。本文是整个《解读2015》系列文章的精简版，整理自郭蕾在ArchSummit大会上的开场演讲。

云计算

过去一年是云计算蓬勃发展的一年。在这一年里，公有云服务竞争更趋激烈，国内外市场均出现洗牌迹象。国内云计算厂商开始进军海外市场。但一年来，国内比较有影响的云安全事故多有发生，企业对使用云计算的信心尚有待提高，重点行业私有云、混合云需求继续走高。我们预测随着云计算产业规模的继续增长，明年的市场将出现分化。大型、超大型客户业务的云化将导致云厂商梯队的重组。新技术的热潮将走向更加务实的阶段。

容器

容器的应用已经开始爆发式的增长，国内越来越多的公司，包括传统公司都已经成为容器技术的早期使用者，容器的普及速度将会比过去的虚拟化等技术快很多。Docker Engine 已经成为整个容器技术栈的基础，社区以及企业开始将注意力往编排和调度的工具上转移。OCI 和 CNCF 出现的恰到时候，他们将共同推动整个社区向着共同目标齐头并进。Docker 有着先入为主的优势，在整个社区有很高的认可度，过去一年，他还在努力做着自己的生态，从引擎到编排再到容器云。

移动

移动开源蓬勃发展，移动应用架构越来越复杂，React Native 带给移动开发者莫大的冲击，Swift 语言开源受到热捧，Apple Watch、TV 和 Android Wear 也给开发者带来了新的可能。Android 系统用户体验大幅度提升，生态系统开始走向多元化。大量高深技术出现，开发者社区更加成熟。智能硬件借助 Android 开始发力，未来的 Android 将会通过物联网改变每个人的生活。

NLP

得益于深度学习算法的快速进展和大规模社交文本数据以及语料数据的不断积累，自然语言处理技术有了飞跃式的发展。在这一年，各大厂商致力于解决语音识别、语义理解、智能交互、搜索优化等领域更加复杂、困难的问题，持续不断地对原有产品的算法、模型进行优化与革新。在这一年，既有大型 IT 巨头以加强研发、扩大合作的方式试图占领自然语言处理领域的半壁江山，也有富有创造力的新兴黑马带着具有竞争力的产品不断涌现，试图与巨头们一较高下。

运维

伴随云计算的火热，运维显得尤为重要，DevOps、NoOps、SRE 不断地被诠释、应用、

尝试。在大规模、复杂架构的催生下，自动化运维站到一个风口，她给传统企业带来了福音，给基础运维带来了巨大的挑战与机遇，同时也给越来越多的企业带来了新的抉择。容器技术对服务部署和运维带来革命性的影响，在未来的一年，也将是运维容器化的元年。

整体来看，在过去的 2015 年，开源得到了飞速发展，各个技术领域的基础设施基本都是开源软件，开源也加速了行业的发展。喊了多年口号的云计算，也逐渐找到了落地点，国内外的各大企业都已经完成相关布局。同时，随着云计算的普及，运维领域发生了翻天覆地的变革，运维人员以及企业运维的定位也与之前完全不同，所以有人开始提倡『运维 2.0』。容器技术自 2013 年被 Docker 引爆以来，一直持续走火，到现在，国内已经有数十家基于容器构建的云服务商 .....

去年我们在策划『解读 2014』系列文章时，我们的总编辑崔康就提到说，整个 IT 技术领域发生了许多深刻而又复杂的变化。然在今年年底我再去回顾的时候，发现这句话还适用，也许技术的发展本身就是这样。希望“解读 2015”年终技术盘点系列文章，能够给您清晰地梳理出技术领域在这一年的发展变化，回顾过去，继续前行。

CONTENTS / 目录

行业篇

技术篇

解读 2015：IT 职场发展

作者 董志南

1. 概述

又到 2015 年年底，回顾这一年，IT 圈发生了很多有意思的事，“大众创业、万众创新”提升到了一个新的高度，尽管在整体经济形势不好的情况下，IT 公司的发展却没有受到大的影响，创业公司如雨后春笋似地发展。与此同时，国内很多领域的巨头都开始合并，比如携程和去哪儿网，美团和大众点评，滴滴和快的，这也说明了另外一个现象，行业内的竞争非常残酷，大家也都开始求同存异，谋求共同发展。在这样的背景下，2015 年，IT 人的就业形势发生了怎样的变化，各个技术领域又有怎样的跌宕起伏呢，2016 年的发展趋势会是怎样呢？针对以上问题，近期，InfoQ 联合著名 IT 行业招聘网站：拉勾网，共同发布了《2015 年 IT 职场发展白皮书》，为您共同揭晓上述问题的答案。

本次报告基于拉勾网提供的 10+ 万家 B 端互联网公司数据，以及 400+ 万互联网从业人员中的 200 万名程序员数据编制而成。

2. IT 行业整体发展情况

2.1 薪资待遇

进入 2015 年，“互联网 +”成了最火热的词汇，一方面传统行业对 IT 技术的态度从“被动”转向“积极拥抱”；另一方面，IT 技术几乎渗透到了人们生活的各个领域，这一趋势不可逆转。如此一来，本来就很紧张的行业人才市场，随着众多传统行业的进入，使得跨界人才争夺首先在 IT 行业出现。

- IT行业平均薪水当属帝都最高。

随着国家提出“大众创业、万众创新”的号召，使更多的人投入到以互联网为代表的 IT 行业，这自然引起了行业人才“价格”的水涨船高。在 2014 年全行业普遍薪酬增长在“<8%”时，2015 年整个 IT 行业的薪资发展情况怎么样?

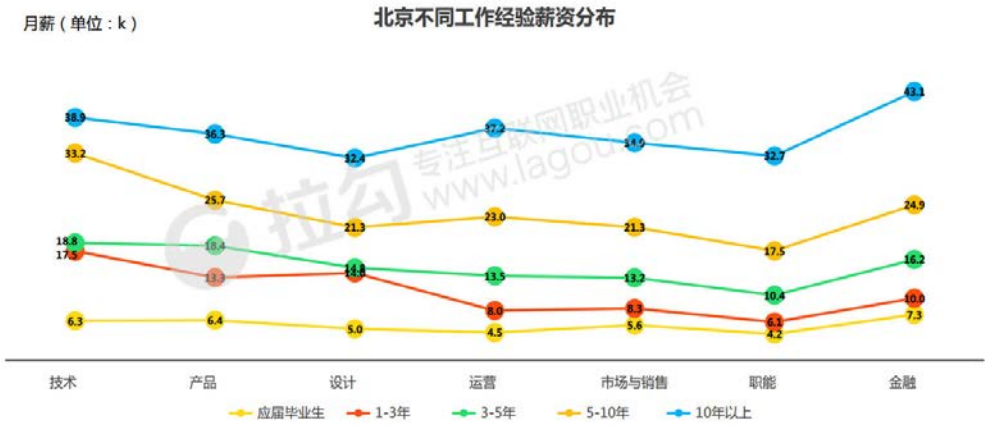
根据拉勾网提供的数据显示，2015 年，整个 IT 行业的薪酬水平可以用“高薪”二字来概括!

从地域上来看，全国范围内，IT 行业发展最好的六大城（北京、上海、深圳、杭州、广州和成都）中，平均月薪当属北京最高，达到 12.4K/ 月，可见作为 IT 行业发展的“国家队”，北京的 IT 从业人士收入明显处于领先位置；紧随其后的上海、深圳分列第二、第三名，分别为 12K/ 月和 11.2K/ 月，然而同样作为国家大力发展的超一线城市，尽管消费水平与帝都不相上下，但 IT 从业人员的收入水平仍旧难以与帝都抗衡；值得注意的是，同样作为超一线城市，生活成本与消费水平与北京、上海处于同一水平线的广州，其 IT 从业人员的平均收入水平仅仅为 9K/ 月，仅排在六大城的倒数第二位，仅比以“生活安逸、消费适中”而著称的宜居城市——成都高 900 元 / 月。

从岗位上来看，产品岗和技术岗仍旧是 IT 行业的王牌岗位，薪资领跑行业内的其它岗位。随着 IT 技术的不断发展以及各个厂商对于产品功能设计、用户体验重视程度的提高，产品岗位和技术岗位已经当之无二地成为 IT 公司内的核心岗位，薪水也分列一、二名，分别为 14.46K/ 月和 13.24K/ 月，以绝对优势远超出金融、设计、市场、运营和职能岗位。

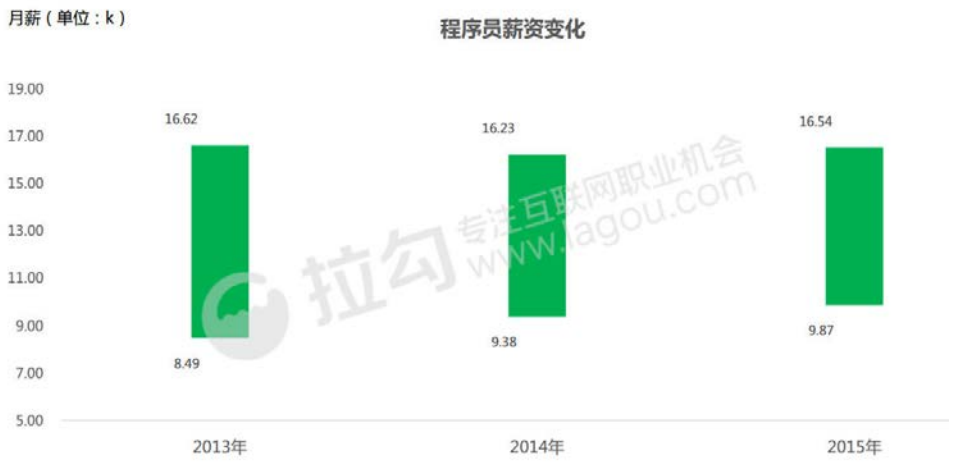
随着工作年限的增长，工作经验与人脉、阅历的不断积累，薪资水平自然水涨船高。根据拉勾网提供的数据来看，IT 从业者的月薪大约是以每年 2000 元的速度进行增长，其中 3 年工作经验是一个十分重要的“分水岭”，工作经验低于 3 年，薪资水平涨幅较为迟缓；工作经验高于 3 年的，薪资水平有了大幅度的提升。值得注意的是，工作经验为 10 年以上的人，薪资水平将会呈现出“跨越式”的提升，可见具有 10 年工作经验的资深工程师 / 管理者，无论是在技术经验上、管理经验上、还是职场阅历上，都要远远强于工作经验较少的人，使得企业不惜开出昂贵的薪酬留住人才。

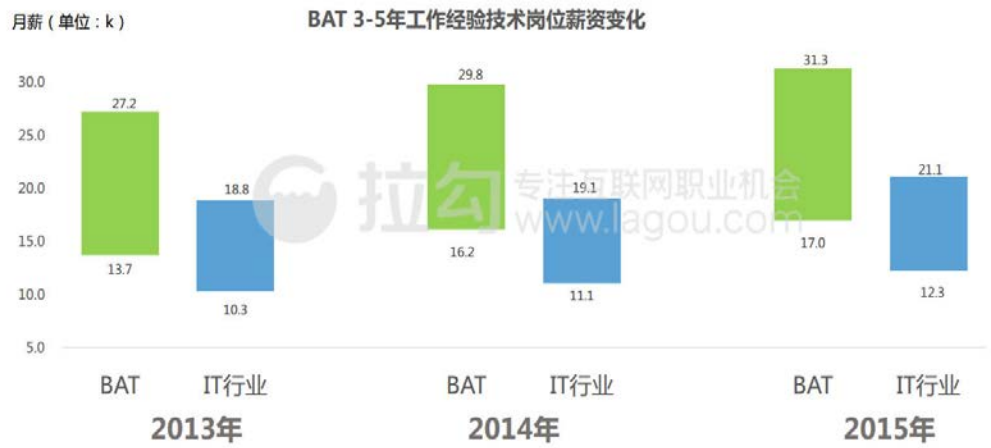
仅以帝都为例，如下图所示，无论是什么岗位，随着工作年限的增长，薪水自然水涨船高。但就单一类型的岗位来说，技术类岗位与设计类岗位中 1-3 年工作经验的人与 3-5 年工作经验的人就薪资待遇上来说并无明显提升；技术类和设计类人员在第 5 年往后，薪水有了质的飞跃，而金融类岗位的人要到第 10 年往后，薪水才能有较大幅度的提升。在刚踏出校园的阶段，从业人员往往处于“一张白纸”的状态，无论从事何种岗位来说薪资都处于较低水平，而随着技术与经验的积累，薪资有了较大的提高，1-5 年之内，对于技术与设计岗位的人员，他们往往还处于一线岗位，薪资提升幅度不大，而 5 年过后，大部分从业人员已经走上了管理岗位，薪资往往有了“跨越式”发展。而金融从业人员在前 10 年，薪资都是处于逐年稳步上升的态势，第 10 年过后，个人的从业经验到达了顶峰，薪资自然有了质的飞跃。然而，IT 公司中的运营、市场、销售、职能岗位，由于往往处于非核心岗位，起薪较低，随着工作年限的加长，薪资也呈现稳步上升的态势，具有 10 年以上工作经验的人员，其薪资待遇可与技术研发、产品设计等核心岗位相抗衡。



对比 2015 年、2014 年与 2013 年的数据，我们发现，程序员的最高平均薪资基本保持平稳，但最低平均薪资却在逐年增长，分别为 8.49K/ 月、9.38K/ 月和 9.87K/ 月。可见，随着 IT 公司对于技术人才需求的不断旺盛和盈利模式的不断革新，IT 技术人的收入也有了一定幅度的上调。这一方面预示着 IT 行业仍然处于高薪行业的行列之中，另一方面也在整体经济下滑的今天为广大勤勤恳恳、苦心钻研的 IT 技术人员打了一针强心剂。

对比各个公司的情况来看，IT 行业待遇最好的公司非 BAT 莫属。如下图所示，以 3−5 年工作经验的程序员为例，BAT 三家巨头的平均月薪远远地超过了 IT 行业的平均水平，呈现压倒性趋势。从 2013 到 2015 年，BAT 三家公司 3−5 年工作经验的程序员的平均月薪比 IT 行业平均水平分别高 40.5%、52.3% 和 45.1%。这些数据从侧面表明，当下百度、腾讯和阿里巴巴仍然是 IT 行业当之无愧的领头羊式企业，不但从技术上起着领跑作用，从薪资待遇上也完胜大多数 IT 公司，从而不断地挖掘和收揽更多优秀的人才。



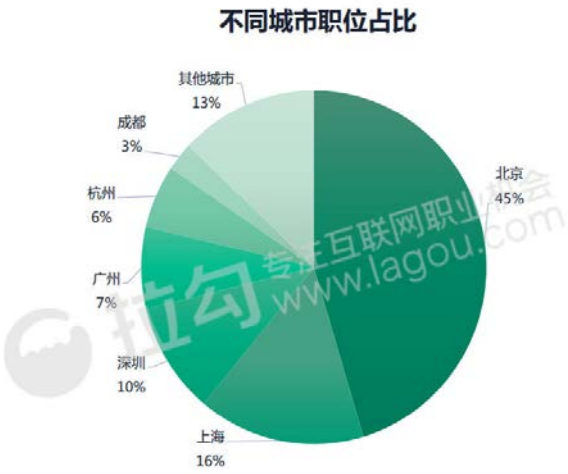


2.2 行业发展

在今年的政府工作报告中，李克强总理提出政府将制定“互联网 +”行动，推动移动互联网、大数据等与现代制造业结合，促进电子商务、互联网金融健康发展。这是此行业热词在历年总理政府工作报告中的首次现身。“互联网 +”第一次纳入国家经济的顶层设计，对于整个 IT 行业，乃至中国经济社会的创新发展意义重大。如今，“互联网 +”正强势发展。O2O、金融、教育等板块一飞冲天，可见 2015 年“互联网 +”行情正在轰轰烈烈地渗透到各个传统行业领域，“+”啥啥涨！

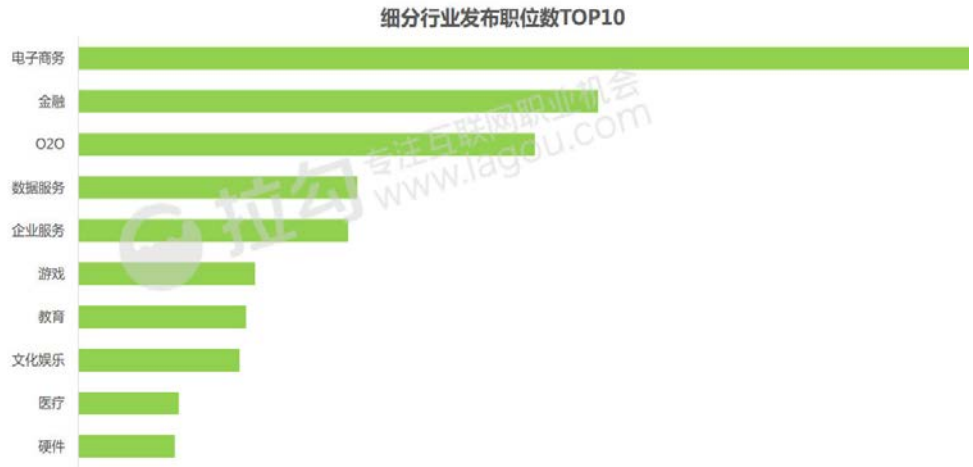
这表明，在“大众创业、万众创新”的当今环境下，IT 行业的创业氛围一片大好，创业公司如同雨后春笋般层出不穷，不但带活了整个国民经济、使得 IT 行业犹如潺潺的清泉，充满活力，更是为广大求职者提供了大量的工作岗位，有效地缓解了整个社会的就业压力，为人才市场的供需平衡增添了一份动力。在行业发展情况中，将着重探讨 IT 公司的地域分布、行业集中趋势、招聘集中趋势、创业趋势等方面的发展情况。

从地域分布上来看，IT 行业大多数公司都集中在北京，上海和深圳作为金融中心和对外开放的港口城市则紧随其后，而作为“北上广深”之一的广州则数量较少。由于北京的政府、教育资源相对集中、又有“中关村”这一国家级 IT 创新摇篮坐镇，这也从侧面奠定了北京作为全国 IT 大城的地位。



借助“互联网 +”的东风，信息技术已经逐渐深入到传统行业的方方面面，从而带动了各个行业的飞速发展，例如电子商务、金融行业、教育行业、医疗行业等。在这些行业中，不仅有着大量的初创公司纷纷出现，更带来了大量的就业机会，为早已饱和的人才市场带来一些慰藉。近年来，电子商务、互联网金融和 O2O 成为 IT 行业的三大热门招聘领域，其职位需求占到 2015 年 IT 行业职位需求总数的 60% 以上。现如今网络购物早已变成人们的生活习惯，火热的电子商务更是以压倒性趋势将其他行业 PK 了下去；随着移动技术的飞速发展以及人们理财观念的不断革新，互联网金融也成为 IT 行业的新贵，颠覆了传统的消费与理财方式。

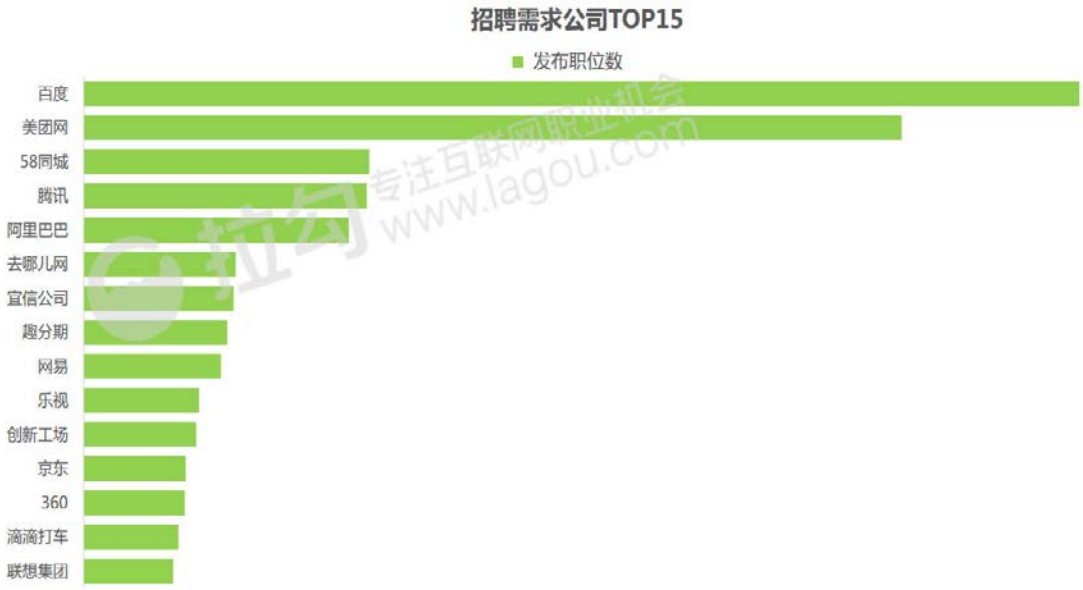
与 IT 行业职位需求相对应，2015 年，行业职位增长最多的 Top5 也分别为电子商务、金融服务、O2O、数据服务和企业服务。该数据表明电子商务、互联网金融、O2O 以及 IT 服务行业无论从职位需求的数量上还是增长速度上，相较于其他领域都呈现压倒性趋势；而行业职位增长最缓慢的 Top5 则分别为广告营销、旅游、安全、社交和招聘行业，虽然这些行业曾经火爆一时，但是现阶段逐渐增速放缓，呈现遇冷态势。尽管从数据上来看，电子商务、互联网金融、O2O 领域的职位需求发展迅猛，但是从今年下半年开始，这些领域则经历了前所未有的“资本寒冬”，一批又一批的 O2O 企业纷纷倒下，成为这轮“烧钱大战”中的炮灰，众多互联网金融公司也风光不再，难以返回去年的风光模样。



行业增长速度以15年比14年新发布职位多少来计算

自从 2015 年初，IT 领域可谓风起云涌，各个 IT 公司纷纷开始转型，不断拓展行业渠道。反映在就业市场上，IT 行业的招聘需求呈现“井喷式”增长。从 IT 公司的招聘需求来看，2015 年，大公司的招聘需求仍旧高昂。其中百度以压倒性趋势荣登 2015 年度招聘需求 Top1，紧随其后的是业内发展如日中天的 O2O 公司——美团。而 BAT 三驾马车中的腾讯和阿里巴巴似乎在 2015 年这个经济低迷的行业寒冬略显疲态，招聘需求仅排在第 4、5 位，连百度的 1/3 都不到。自从三巨头们嗅到了行业寒冬的意味，纷纷缩减了招聘需求，阿里巴巴大规模缩减校招，百度宣布暂停了社会招聘，腾讯也宣布暂停外包招聘了，不知黎明前的第一缕阳光何时才能到来。

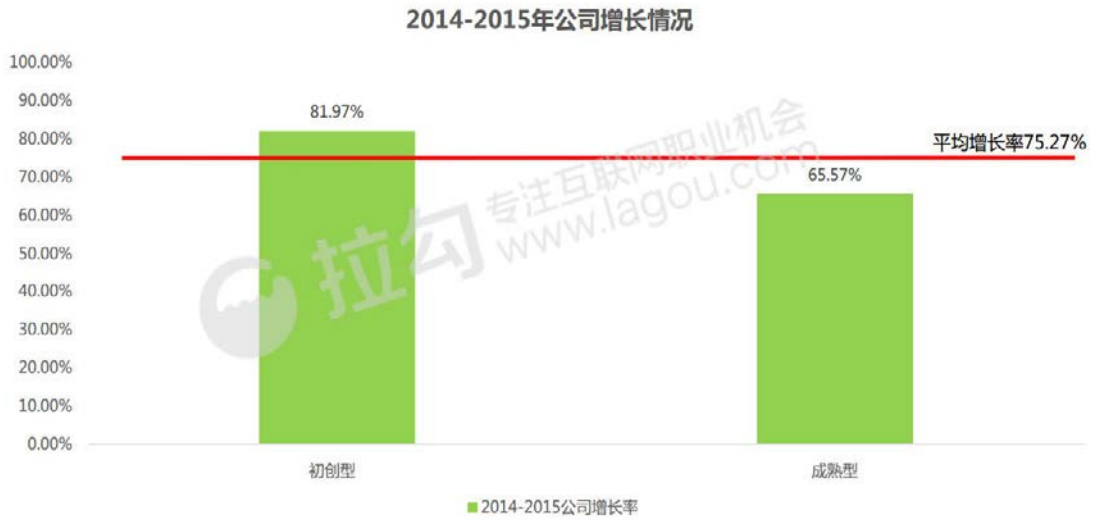
然而，无论是资本寒冬还是帝都雾霾，创业者们的热情依然无法抵挡。今年的政府工作报告中，两次提到“大众创业、万众创新”。众多创业投资机构也在今年纷纷涌现，而互联网领域的创业则尤其明显。一方面由于政策与创业环境迎来重大利好，为创业者提供了不可多得且相对公平的机会；另一方面也由于 VC 投资正在进入高速增长期，互联网创业领域被热捧，融资更加容易；移动互联网顶层开发环境与上游配套设施的成熟，降低了移动互联网创业的门槛，这也促使了有好的想法与执行能力的年轻人带领团队进行创业；另外，大公司庞大的组织机构导致员工的自我价值受限，对于掌握了资源、人脉或技术的互联网大公司的员工，往往更容易“出走”，利用自身的技术和人脉从事创业。这些现象共同使得互联网创业成为了一种集体行动。2014 年至 2015 年间，创业型



公司层出不穷，为 IT 行业寒冷的冬天增添了些许春意。其中，新增的初创型公司较 2014 年增长了 81.97%，新增的成熟型公司较 2014 年增长了 65.57%，新增公司总体上较 2014 年平均增长 75.27%，IT 行业内公司的增长主要由创业型公司带动。

从行业角度来看，2014 年至 2015 年间的创业公司中，“安全”、“O2O”、“移动互联网”以及“互联网 + 医疗”4 个领域发展迅猛，创业公司的增长率分别为 163%、132%、107% 和 101%；而之前业界一致呼声很高的“互联网 + 教育”、“互联网金融”、“电子商务”，其创业公司的增长速度则逐渐放缓，呈现疲态。由于近些年来，大家越来越意识到网络安全的重要性，尤其是随着云计算、云存储技术的不断发展，网络安全成为了互联网生态链中越来越重要的一环，自 2014 年起，网络安全初创公司成为了硅谷风投界的新宠，无疑也对国内的创业方向产生一定的影响；而尽管不断有各垂直领域 O2O 企业宣布获得融资，一批又一批新兴的 O2O 公司前赴后继地扑向这个白热化的“烧钱大战”当中，但目前业界对于 O2O 行业的“泡沫化”和“竞争过于同质化”不甚乐观，更有专业人士称“O2O 行业面临倒闭潮，发展不可持续”；互联网在线教育突破时间空间的限制，狠狠的击中了优质教育资源的共享，节约路程时间等用户痛点，

一度成为 2014 年互联网融资热炒的焦点，然而以“龚海燕梯子网资金链断裂”事件为标志，资本市场的转冷，让没有明确的盈利模式的互联网在线教育的创业在 2015 年增速放缓。这也就不难理解网络安全领域火热发展，而 O2O 领域、在线教育发展放缓的原因。



IT 技术高速发展的条件下也带来人才的高速流动，据拉勾网提供的数据可以看到：应届毕业生是流动性最高的一族，由于职业定位尚未成型，可塑性相对较高，平均不到半年跳槽一次；工作 3 年以上的人群，个人的能力和经验有了一定的积累，出于更好的职业发展或是更加稳定的工作环境的诉求，1–2 年也会跳槽一次；而工作 10 年以上的人群，由于各个方面均已成型，在企业中往往担任高层岗位，对于薪水待遇、工作环境、职业发展路径均有了明晰的定位，往往不容易跳槽。

2.3 年龄、性别与学历

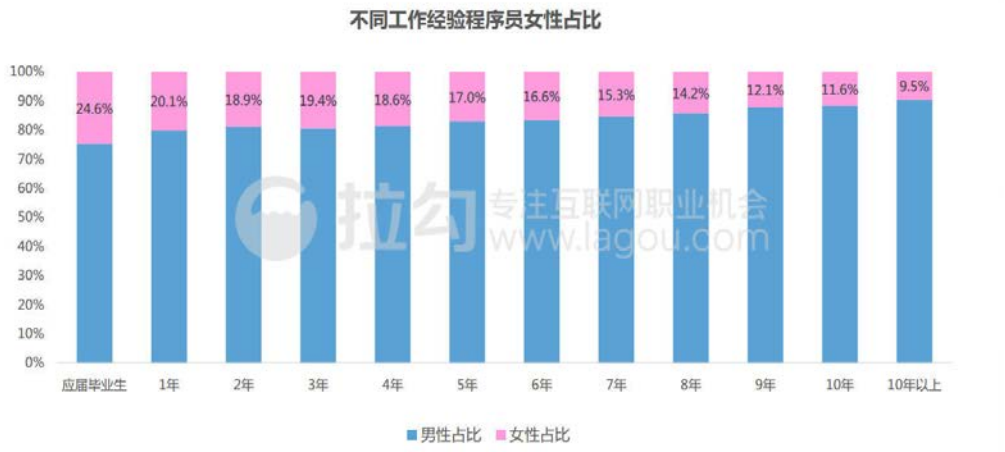
2015 年 12 月 10 日是世界上第一个“程序媛”——Ada Lovelace 诞辰 200 周年。虽说第一个程序员是女性，但如今 IT 行业早已被男人所“主宰”。甚至著名的 Bloomberg 都给男性程序员起了一个又酷又时尚的名字：“Brogrammer”，一改往日程序开发人员呆板极客的形象，这也凸显了 IT 行业以男性为主导的发展方式。Datamation 的统计数据显示，美国只有 1.5% 的开源项目开发者是女性，可见女性程序开发人员在整个 IT 行业中十分“罕见”。那么 2015 年，整个 IT 行业中，女性程序员的从业比例如何，相比去年有了怎样的变化呢？

根据拉勾网提供的数据显示：2015 年，女性程序员的占比为 19.62%，2014 年，女性程序员的占比为 19.43%。从男女比例上来看，2015 年和 2014 年基本持平，女性 IT 人员占总人数的比例均保持在 19.5% 左右。说明在整个 IT 行业仍旧呈现严重的“重男轻女”现象，男性从业者还是主导了“大半壁江山”，整个行业呈现明显的“阳盛阴衰”的态势。

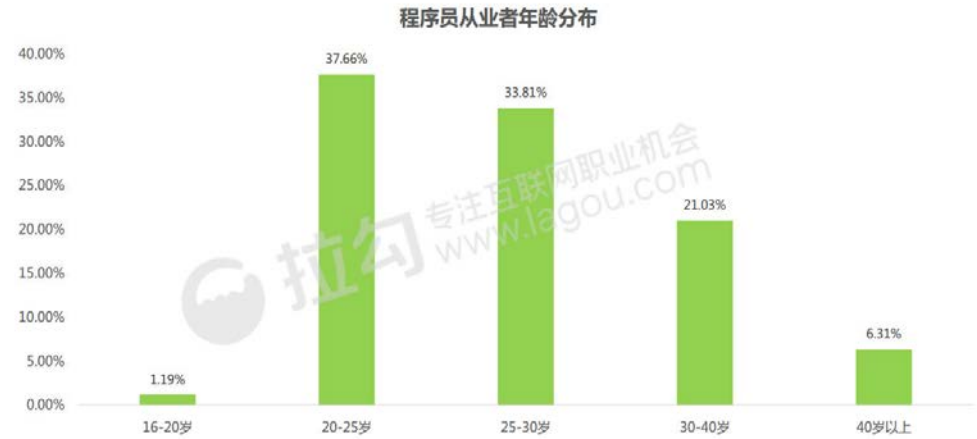
整个 IT 行业呈现明显的“男多女少”的状态，但最严重的还属技术岗位。在技术研发岗位中，平均每 5 个程序开发人员中才有一个女性。而市场与销售、运营、设计和职能等岗位随着 IT 技术能力要求的降低，沟通交流能力以及组织策划等综合能力要求的提高，女性从业人员的比例不断上升。尤其是职能岗位，由于主要是行政、人力资源、财会等岗位，而这些岗位往往是女性较为青睐的岗位，因此呈现男女比例较为均衡的迹象。

在程序员中，随着工作年限的不同，女性所占的比率也不尽相同。从拉勾网给出的数据可以看到，随着工作经验的增长，女性程序员的流失率也越来越大。可能是由于随着职场经验的不断积累以及职场人脉的不断丰富，尤其是自我职业定位的不断明晰以及家庭、性格、个人爱好等多方面因素，女性程序员逐渐逃离这个相对辛苦的工作岗位，转而奔向相对稳定轻松的非技术岗位。

而根据拉勾网提供的数据显示，随着年龄的增长，针对“程序开发岗位”求职的人员呈现明显的“阶梯状”分布。20−25 岁的年轻程序员占据 37.66% 的人数比例，随着年龄的增长，程序员的人数比例不断降低；30−40 岁的程序员相较

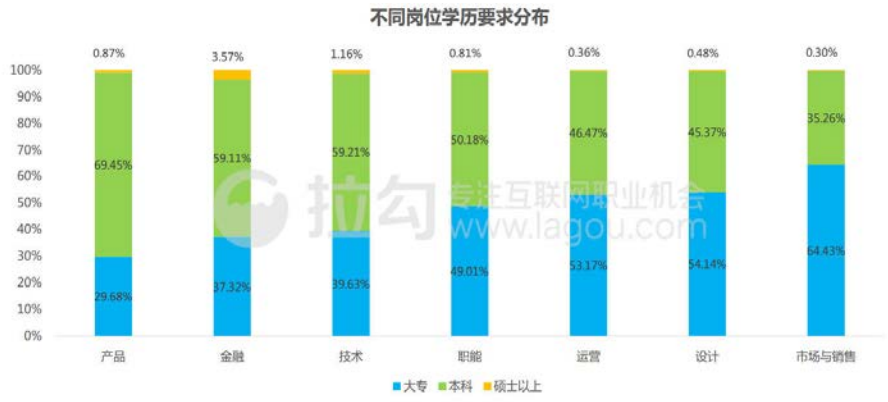


20 多岁的程序员来说，人数上少了将近一半，仅占总人数的 21.03%；而 40 岁往上的程序员所占人数比例微乎其微，仅占 6.31%。可见程序开发岗位往往呈现“年轻化”的趋势，年龄较大的人往往不会再寻求较为辛苦的程序开发工作。一般来说，中国的程序员大都是吃“青春饭”的，大部分程序员的黄金时代是 24 ~ 28 岁。到了 30 岁左右，一批又一批年轻程序员会带来巨大的竞争压力。首先由于 IT 行业的飞速发展，很多自己以前学的东西逐渐升级换代，而许多程序员由于年龄增长，学习新知识的效率必然下降；其次工作几年后，薪水要求自然升高了，而年轻程序员工资又低、干活又快，当然会成为企业的首选；第三，30 岁的程序员基本成家，要支撑家庭的生活负担，几乎没有了连从头在来的勇气。中国的老话说“长江后浪催前浪、一代新人换旧人”，这个历史的规律在 IT 开发行业体现的尤其明显和残酷，很多程序员如果无法转岗，必然要面对的结果就是降薪乃至失业。



一直以来，IT 行业总是作为一个“神秘的”、“高精尖科技”的行业存在于人们的视角当中，但似乎 IT 行业这个“高科技”行业对于求职者的学历要求貌似没有那么的高，它从没有把“高学历”、“名校毕业”作为严格的门槛对求职者进行把关。在大家熟知的 IT 大佬中，既有搜狐总裁张朝阳先生这样的，身披麻省理工学院博士后光环，名校毕业、学历高的令人却步；也有阿里巴巴董事局主席马云先生，毕业于杭州师范学院（现已更名为杭州师范大学）、非 IT 专业出身的“小本科”。似乎 IT 行业从不缺海外名校毕业，拥有漂亮教育履历的人担当主力，但也常常有专业、学历不是那么抢眼的人，却也能在惊涛骇浪的 IT 大潮中翻云覆雨。那么，IT 行业这样一个对于技能要求如此之高的行业，对于从业者的学历要求到底是怎样呢?

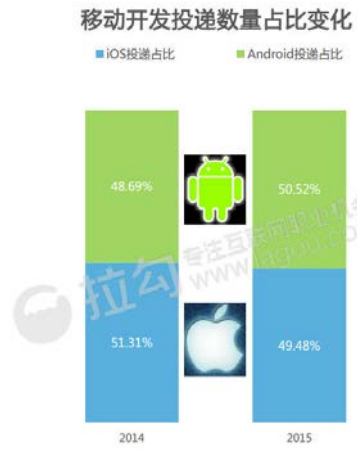
虽然 IT 行业对于人才需求的专业性较高，但是对于学历的要求却不是硬性的。根据拉勾网提供的数据来看，IT 行业中所有岗位类型对于从业人员的学历要求都集中在大专和本科，极少数的岗位会由硕士以上的人员担任。其中，产品类岗位对学历要求最高，超过 70% 的职位要求本科以上学历，市场与销售岗位要求最低，仅三分之一的人员要求有本科学历。由于产品岗位需要具有一定的业务背景知识、较强的学习能力和用户心理把控能力，因此对于学历要求稍高；金融岗位需要较高的专业背景知识，因此硕士以上学历的人员比例相对较高；随着IT培训的遍地开花,即使学历不高,只要具有浓厚的兴趣和刻苦学习的精神，通过大量的自学和项目锻炼，也可以胜任技术研发岗位，因此技术岗位的学历并非“非本科不可”；而对于人的综合素质（尤其是组织协调、与人打交道的能力）要求较高，相对来说对于学历不做严格限制的市场与销售岗位，本科及以上的人员所占比例则最低，仅为 35.56%。



2.4 技术岗位变化

随着 4G 网络的覆盖范围的不断地扩大，未来的移动通讯协议还在向着更快的 5G 进军，移动 App 早已经满足用户各方面的个性化需求，从衣、食、住、行到游戏、社交、阅读，几乎任何资源我们都能在提供移动 App 下载的电子市场找到，App 化已成为目前移动互联网的最大特点。移动互联时代的迅猛发展使得移动开发人才的需求得到爆发性增长，供不应求所催生的“人才荒”已经成为众多公司发展的重要阻碍。因此，移动开发人员一度成为行业内招聘的“新宠”，移动开发岗位也一度成为热门岗位。那么，最近两年，移动开发的求职情况又有什么样的变化呢？

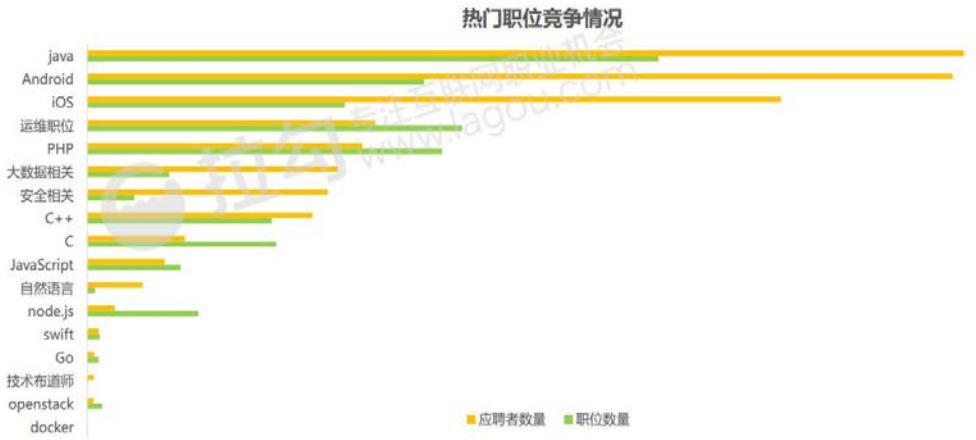
从拉勾网给出的数据来看，相较于 2014 年，2015 年 IT 行业内 Android 开发的职位数量略有减少，iOS 开发的职位数量略有增加，但二者基本上还是和去年保持持平；反过来，2015 年 IT 行业内 Android 开发的简历投递数量略有增加，而 iOS 开发的简历投递数量略有减少，二者也基本上还是和去年保持持平。苹果公司的产品凭借其美观大方的外形设计、优秀流畅的使用体验一贯受到人们的热衷与喜爱，并且 2014 年 iPhone 6 以及 2015 年 iPhone 6S 等大屏产品的推出，其友好的界面尺寸和精致的外形设计更是得到一大批狂热粉丝的青睐，越来越多的手机应用优先选择 iOS 操作系统进行开发也并不奇怪，由此也催生出诸多 iOS 开发岗位；而 Android 操作系统由于其源代码开放，企业个性化定制更加自由，简单清晰的应用发布过程，以及诸多手机厂商的硬件平台支持，使得 Android 也成为移动应用开发者的热门选择平台。



就移动开发而言，不同的开发经验与职场受欢迎程度也是不同的。如拉勾网给出的数据所示，开发经验与职场上受欢迎程度呈现较为明显的“中间高、两边低”的分布态势。从图中我们可以看到，经验为 3–10 年的开发者受欢迎程度较高，其中 3–5 年经验的开发人员由于年龄优势，技能学习能力与知识吸收能力相对都较强，而且也具有了相当长时间的实践训练，因此比较受欢迎，而且在开发经验为 6 年时受欢迎程度到达顶峰；而经验为 7–10 年的开发人员随着实战经验的不断积累，往往可以胜任技术难题的攻关任务，在职场中也十分受欢迎；应届毕业生以及 3 年以下的开发人员由于技能不够熟练，受用人单位欢迎的程度不太高；而 10 年经验的开发人员往往寻求企业的管理层岗位，寻求技术研发的岗位往往不占优势。

IT 技术作为一个大舞台，各个垂直细分的技术领域都能在上面占据一席之地。其中既有从出现到现在就一直长盛不衰的经典编程语言，也有随着用户需求、产品理念或是硬件支撑体系不断进步而渐渐淡出人们视线的、曾经火爆一时的 IT 技术，也有最近刚刚火起来的“新潮”技术，都是“你方唱罢我登场”。不同 IT 技术的变化发展也催生不同技术岗位与求职需求，就拉勾网给出的数据来看，目前市面上流行的 IT 技术的具有不同的“供需关系”。其中作为一直以来都很经典的编程语言，Java 已经伴随我们走过了 20 个年头。尽管近几年来大家一直“唱衰”Java，但 Java 仍然是诸多 IT 大公司里主要使用的开发语言，而且时下云计算、大数据的火热，作为开源框架 Hadoop 的编程语言，Java 以其竞争力的流处理能力再次回归到人们的视野。正因为如此，也使得 Java 荣登应聘者数量的榜首，成为应聘者学习和使用最多的语言。而伴随着移动技术的火爆，Android 开发和 iOS 开发也具有非常多的应聘者来求职。从应聘者数量和公司职

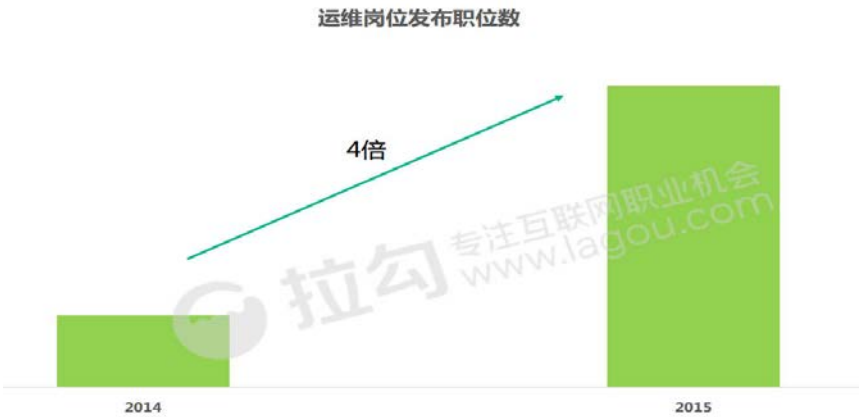
位数量之间的“供需比”来看，PHP、C 语言和 Node.js 成为最“供不应求”的开发语言，我们发现求职者数量远远少于市场上提供的职位的数量，该领域的求职者明显无法满足公司“求贤若渴”的需求，这可能与时下该领域日益增长的业务需求有关。



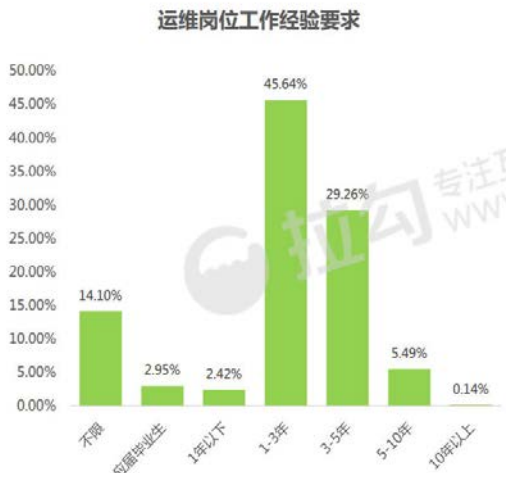
而从各个技术垂直领域的待遇上来看，技术布道师成为当之无愧的高薪技术岗位，以 20.2K 的平均月薪领跑技术岗位薪资榜单。三年前，技术布道师（developer evangelists）还没有出现大众的视野里，直到 2012 年 TNW 的作者 Courtney Boyd Myers 写了一篇关于北美市场上的技术布道者日常生活的文章，大家才开始慢慢对这个技术职位进行了解。技术布道师是最前线也是最重要的“翻译者”，他们能够把技术以易懂的方式解释给来自不同领域的人，以此获得他们对产品或技术的支持。这需要又懂技术又能挖掘出技术背后故事的人才，他们能够激发起人们对于一个产品的激情。而近几年来，技术布道师一直供不应求，主要由于技术布道师专注在开发者技术和营销推广之间的重叠领域，其实这类角色的工作职责在科技公司已经有了，只不过现在以一个更加具体的名字出现；初创公司越来愈多，所涉及的范围也越来越广，很多专注于 SaaS（Software as a Service）和 PaaS（Platform as a Service）的公司出现，平台化的流行间接增加了对技术布道者的需求；技术布道者带有一定的光环，颇具诱惑力——人们发现这个职位的多样性和丰富的出差机会。正因为这样的原因，使得“技术布道师”成为市场上供不应求的抢手人才，帮助公司树立自己的技术品牌形象，获取不同领域（尤其是投资人）对于本公司的技术实力认可，从而拉拢更多的人对于自己产品或技术的支持。

随着智能硬件与个人助理式 APP 的崛起，以及容器、大数据等领域的火热，IT 公司们开始加大布局这些垂直领域的技术研发投入，自然语言、Docker、大数据、OpenStack、安全的职位薪资自然随之水涨船高。而传统的编程语言，例如 C++、Java、PHP 等对应的岗位薪资待遇则处于“不温不火”的状态。

运维岗位的薪水则最低，平均月薪仅有 10.7K。然而尽管如此，2015 年，企业对于运维人员的需求却相比 2014 年增长了近 4 倍，如下图所示。如今 linux 行业崛起，在云计算大环境下，市场上对 linux 运维人员的需求越来越大。而运维岗位又是一个“经验活儿”，它需要的不是“天赋异禀”而是“经验丰富”。是否能够掌握常年积累下来的运维流程、思想和经验体系并非一朝一夕之事，因此高级 linux 运维绝对是目前大公司的稀缺人才。而从运维岗位对于工作经验和学历的要求也能看到，运维岗位对于 1-5 年工作经验的求职者更加青睐，甚至可能具

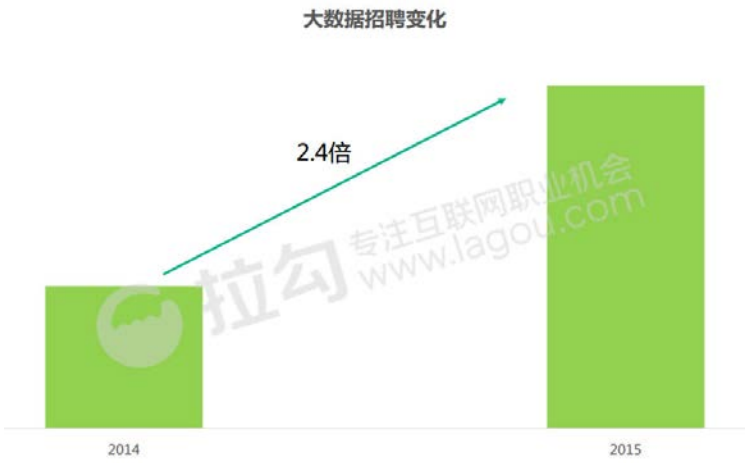


有 5 年以上工作经验的运维高手往往掌握着企业大量运维资料和项目经验，不会轻易跳槽，企业也不会轻易放人，对于投递人员的学历来说也以本科和大专为主，这更加说明运维岗位对于工作经验的高度依赖性。与之相反的是运维岗位的平均薪水却是技术岗位中最低的，这也可能是由于运维、技术保障等领域的技术价值并没有得到相应的体现和认可，这种技术价值与所谓公司价值观倒挂的现象在运维领域尤其突出。



从拉勾网的运维工程师职位描述分词中，我们可以得到运维工程师招聘的常见要求技能：熟练的 Linux 命令操作和维护，良好的服务器配置技能，良好的互联网知识和网络通信知识，能熟练地编写脚本（Shell 或者 Perl），熟悉网络规划、实施方案、服务器及网络设备性能指标等等。

受经济全球化和全球信息化、人类社会发展和需求多样性、云计算和物联网技术深化应用等方面的影响，刚刚过去的一年，“大数据”（Big Data）已经成为 IT 领域和互联网上反复提及的热词，人们用它来描述和定义信息爆炸时代人类社会各领域产生的海量数据，以及大数据时代的来临。而与之相关的职业需求也呈爆发式增长，大数据职业的相关人才匮乏，人才缺口非常大。根据拉勾网给出的数据来看，从 2014 年到 2015 年，IT 行业关于大数据的岗位需求增长了 2.4 倍，这也进一步说明了随着大数据趋势引起越来越多的重视，各大企业对与大数据相关高端人才的需求也越来越紧迫。这一趋势，也给想要从事大数据方面工作的人员提供了难得的职业发展机遇。



而从大数据岗位的工作经验要求和投递人群学历中我们也可以看到，大数据岗位对于工作经验要求较高，更偏爱 3–5 年经验丰富的工程师；本科和硕士学历的求职者也更加偏爱大数据的岗位。这是因为大数据和传统数据的最大区别在于，它是在线的、实时的、规模海量且形式不规整的数据，无章法可循，因此“会玩”这些数据的人就需要深厚的背景知识，才能在海量的“非结构化”数据中“玩”出数据的商业价值，让数据变成生产力。据了解，一名专业的大数据分析师往

往需要报考相关的资格证书，除了证书，其实际开发能力和大规模的数据处理能力是作为大数据分析师的一些必备要素，因此对于业务知识良好的理解能力、

丰富的行业经验和洞察力、对于数据敏感的能力、优秀的统计学技能和社会学技能以及数据整理、可视化和报表制作技能则成为决定一个人是否是优秀的大数据分析师的评判标准。而丰富的工作经验和良好的学历背景可以为这些优秀的素质做有力的支撑，这也就不难理解为什么大数据分析师对于工作经验要求较高，也更加受高学历人才的青睐了。

从拉勾网的数据挖掘岗位的职位描述分词中，我们可以得到大数据工程师招聘的常见要求技能：熟悉各种数据挖掘和机器学习的算法、熟练掌握 Hadoop 和 Python 的开发、熟练使用 SAS、SPSS 等数据统计挖掘软件、具有海量数据处理、大规模数据库、分布式计算的经验、计算机、数学或统计学专业背景等等。

3. 总结与展望

我国已拥有世界一流的 IT 产业，IT 技术具有强烈的渗透性和持续高速发展的内在需求，这使得 IT 技术必然要寻求与传统产业融合，催生新的发展空间。IT 技术 + 传统产业，孕育出 C2B、工业互联网，创新制造业发展路径，构筑智能制造的关键基础；IT 技术 + 商贸金融，孕育出 O2O、移动支付、互联网金融，使虚拟空间与现实空间的融合更加紧密；IT 技术 + 生活服务，孕育出网络社交、在线教育，创新百姓生活方式、提高人民生活质量。展望 2016 年，“互联网 +”的概念将被赋予新的更广阔内涵，IT 技术将加速从生活工具向生产要素转变，IT 技术与传统产业的结合更加紧密、以信息技术为基础的新兴业态更加密集涌现，引发互联网产业、信息技术产业乃至整体经济发展的加速创新与发展。

解读 Android：横向扩张、平稳发展

作者 郭亮

Android 经历了大概 7 年的野蛮成长之后，2015 年的 Android 应该是平稳发展的一年。从 Umeng 最新的报告能够看出国内 Android 设备已经占到了 62.3%，大概是 iOS 的两倍，毫无疑问 Android 已经稳稳的坐到了手机操作系统第一的宝座。如今整个 Android 生态链已经形成，几乎每一个细分的领域都有相应的厂商，已经很难挖掘到新的空白地带，无论是 App 的种类、开发工具、分发渠道、测试平台等等都已经初步形成了规模。如今的 Android 已经度过了适应市场、自我完善的时期，如果说 2015 年之前的整个 Android 生态拼的是眼光与运气，那么 2015 年的 Android 拼的就是实力。本文从系统与生态环境、技术与社区、智能硬件三个方面总结了 Android 在 2015 年的发展情况，并且在文末展望了 2016 年 Android 的发展方向。

系统与生态

系统

2015 年谷歌发布了 Android 6.0（Android M）操作系统。Android M 改进了权限管理功能。可以对相机、联系人、电话、短信、日历、传感器等多项权限进行单独设置，不再像以前那样安装时给个权限列表，而是类似于 iOS，调用时才会弹出提示。同时 Android 6.0 引入了一项大家期待已久的功能——指纹识别。Android 6.0 在系统层面加入指纹识别，提供了原生指纹识别 API，这不但降低了厂商开发指纹识别模块的成本。最重要的是原生指纹识别将会大大提升 Android 手机的指纹识别支付安全性。

6.0 之前的 Android 系统用户体验、系统性能、系统安全等多方面都不及 iOS，但 6.0 之后，这种状况得到了很大的改善。使用搭载 Android M 的手机，有着与 iOS 设备相同的流畅性。很多人感觉 Android 的手机太卡、经常死机，相信随着搭载最新系统的 Android 机出现，这种抱怨会慢慢减少。但碎片化一直是 Android 无法解决的一个问题，虽然谷歌也在积极想办法，但 2015 年这一现象似乎变得更加严重。

生态

2015 年 Android 生态链变得更加完善。软件、硬件、支付、分发平台等多方面都在平稳前进。

Android Pay 已经于今年 9 月份正式在美启动，首批支持该服务的商家将超过 100 万家。国内方面，谷歌与银联合作，正在部署 Android Pay 市场。

分发是 Android 生态中最重要的一个环节，国内的 Android 分发环境很复杂，存在很多不良竞争及黑色地带。Google Play 对于 Android 生态来说意义重大，Google Play 未来会是所有 Android 产品的一个出口，如果没有了中国的几十亿用户，那么无论如何，生态都是不完整的。2015 年 Google 已经为入华开始准备，

下架了大量盗版产品，并且给出了 1 台设备 1 美元的补助。相信不久后便会出现中国版的 Google Play。但面对中国特色，Google Play 想要一统天下，只能说任重而道远。

Android Wear 平台上已经超过 4000 个应用。2015 年 Google 对 Android Wear 多次升级，开放了 WiFi 功能，并且推出了大量表盘。如果智能手表是下一个风口，那么廉价的 Android Wear 必定是赢家。在 2015 年的 I/O 大会上，谷歌还发布了物联网操作系统 Brill，它对硬件要求超低，32MB 内存就能够流畅运行。同时到来的还有物联网通信协议 WEAVE，它可以让手机、Brillo 设备和互联网三者互相连通。硬件方面不能忽略的一个分支便是智能汽车，到 2015 年支持 Android Auto 的汽车品牌已经达到 35 个。和苹果 CarPlay 的 34 个相比棋逢对手。

技术与社区

技术

技术是推动产品前进的核心。Android 的发展，离不开技术的进步。2015 年出现了大量优秀的技术方案。移动产品的 Native 形态注定会带来升级迭代这个痛点，每一个 App 厂商都踩过升级迭代的坑。动态加载、热更新这些被大家期待已久的技术开始逐渐被应用到了生产环境，也出现了大量开源的技术方案。

2015 年不得不提一个重大技术革新是跨平台。虽然之前也有一些跨平台的方案，但因为体验太差，并没有流行起来。由 Facebook 开源的 React Native 的出现，解决了多年来的技术瓶颈。2015 年 Facebook 宣布开源原生应用开发框架 React Native 的 Android 版，从而实现了 React 的彻底开源化。有了 React Native，开发者就可以用一套代码写出运行于 Web、Android 与 Android 之上的 UI，这将大大减少人力与开发成本。并且能够解决更新不及时的问题。移动互联网发展到如今这一阶段，跨平台已经成为每各厂商共同关注的问题。国内关于 React Native 的实践也非常积极。社区中出现了大量 React Native For Android 的教程与总结。国内像天猫这样的大厂商在 React Native 出现之后便成立了专门的研发团队来实践这项技术。

Android 开发技术越来越成熟，2015 年移动架构的概念开始出现。部分国内的公司也开始设立移动架构师这样的职位。移动开发不同于服务器端程序开发，不仅仅涉及到业务，还要处理 UI。传统的 MVC 从开发效率、团队协作等角度来讲，并不完全适合于移动开发，2015 年出现了 MVP、MVVM 这样的移动架构，也有 RXAndroid 函数响应式编程的思想。虽然这些架构也受到了很多争议，但移动架构的出现，代表了移动行业发展的一个新节点。

从 2015 年开始，Android Studio 已经成为了 Android 开发的核心工具，Eclipse+ADT 的时代已经结束。谷歌官方不建议使用 Ant 的方式，大力推广 Gradle 的方式来构建 Android 应用。最近发布的 Android Studio2.0 测试版，带来了开发者一直期待的热更新（Instant Run）功能，一次编译项目后，代码或资源文件的修改可以秒装到手机上，这将大大加快 Android 的开发效率。并且最新的版本中 Gradle 速度也有了很大提升，模拟器性能更强。

社区

2015 年 Android 国内外的开发者社区在都非常活跃。出现了大量的开源项目、技术教程，Google 官方也录制了多个系统的 Android 开发、性能优化教程。以 Facebook 为代表的国外厂商与以阿里为代表的国内厂商，为 Android 社区贡献了大量优秀项目与精品文章。

InfoQ 策划的《Android 周报》从 2014 年开始维护，至今已经产出近 80 期内容。《Android 每周》每期会收录 6 ~ 8 篇精品技术文章，几乎覆盖到了 Android 的每一个技术领域及所有的优秀技术专家博客。从这不到 2 年的时间中，我们能感受到国内 Android 开发者的技术实力在一步一步提升，无论是从技术深度、广度还是技术博客的数量上，2015 年都远远超过之前的每一年。甚至 2015 年我们的技术走出国门，走向国际的舞台。

大公司有财力与人力去推动技术的发展，所以大公司往往是技术的拓荒者与领路人。2015 年我们能够非常明显的感受到这一点。微信团队、淘宝团队、QQ 空间终端开发团队等都推出了移动技术分享博客，其每篇分享堪称精品。

谈到社区环境，最重要的一点是开源，如果一个行业没有开源项目的支持，想要生存下来几乎是不可能的。几年前我们接触的 Android 优秀开源项目大部分来自国外的厂商、大牛，鲜有国内的项目。2015 年国内产生了大量优秀开源项目，有阿里的 dexposed、360 的 DroidPlugin 等，这些项目在 GitHub 上引来了大量的关注。国内的 Android 技术人开始积极的参与开源项目，分享研发成果，很多公司的招聘 Android 开发者的时候，要求有 GitHub 账号，并且参与、创建过开源项目。

智能硬件

从多份年终报告可以看出，智能硬件是 2015 年发展最火热的行业之一。前几年大量财力、人力都投入了到了 App、游戏创业的红海，经过几年的发展，每个方向都形成了初步格局。2015 年，投资人、创业者开始将目光从软件移动到了硬件。Android 系统是开源的，各大厂商都可以基于 Android 去开发自己的硬件产品。一年内出现了大量创业团队进入智能硬件行业，有做医疗硬件产品的，有电视、有教育产品等。涉及的范围也很广，大到汽车、电视，小到手表、手环。

2015 年 Google I/O 大会更新了 Android Wear 的功能，Android Wear 拥有了更加完善的操作方式、更全面的应用支持。由摩托罗拉推出的智能手表 Moto360 二代作为国内发售的第一款搭载官方 Android Wear 的设备，正式在上海发布，标志着 Android Wear 正式落地中国。同时国内的大量 App 已经开始支持 Android Wear。

展望 2016

1. 随着 Android 6.0 的推送，大量新旧 Android 机会搭载 6.0 系统。用户体验的改善加上价格的优势，Android 手机在国内的市场份额还会稳步提升。

2. 移动架构会逐渐成熟，业内会形成一些统一的观点。

3. 大量互联网公司会使用 React Native 来开发 App，特别是大的厂商与传统互联网公司。

4. 智能硬件方面，会出现大量以 Android 系统为基础的不同形态的智能硬件。

5. 虽然 Google 与苹果都看好 Watch 的前景，但结合 2015 年整个智能手表的发展来看，Android Wear 在 2016 年并不会像当年的 Android 一样快速被人们接受，笔者认为 Android 手表注定是一个小众产品。

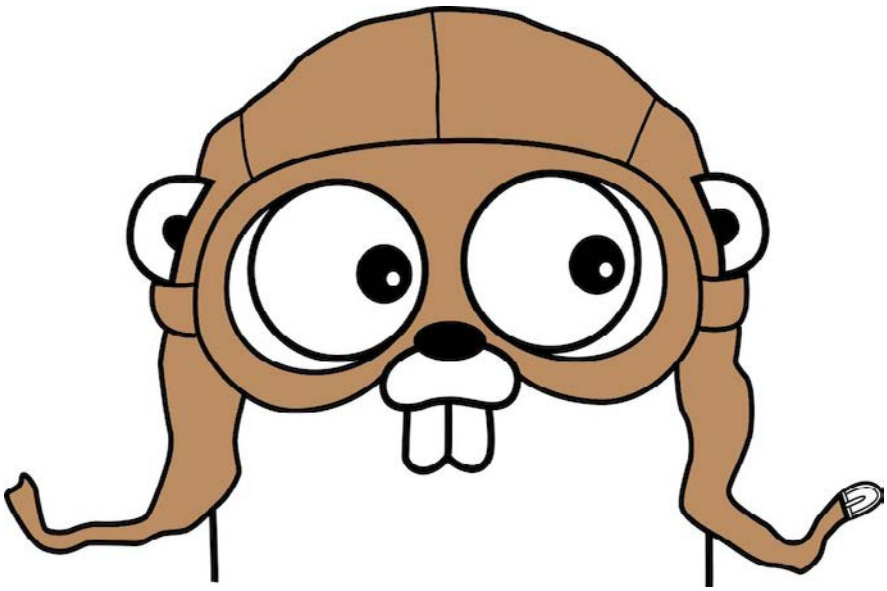
6. 随着中国互联网的快速发展，以 Google 为首的国外大厂商会逐渐进入中国市场。Google Play 中国版会在 2016 年的适当时机、独特的方式进入中国市场。甚至 Google Play 为了获取用户会软硬兼施。

7.2016 年的另外一个战场是移动支付。Android Pay 与银联的合作会成为 Apple Pay、微信、支付宝的强劲对象。相信 Android Pay 会很快分得一块蛋糕。

8.Android 已经发展了近 8 年的时间，短期内出现了大量 Android 从业者。虽然目前前端开发人员并不饱和，但由于今年以来，移动互联网正在经历资本寒冬，2016 年整个行业或许会面临更新艰巨的挑战。Android 开发岗位不会增长，薪资也会处于一个理性的等级。

解读 Golang：Golang 的全进发时代

作者 郝林



现今，21 世纪的第 2 个十年已经过半，互联网也真正进入了极速发展的阶段。在国内， 大家已经对“云计算”和“大数据”等名词耳熟能详了。在互联网软件开发领域，最主流或火爆的技术也无不与之有关。就拿 Golang（也可称为 Go 语言）来说，它就号称“云计算时代的 C 语言”。Go 语言在软件开发效率和运行效率之间做出了绝佳的权衡。这使得它既适应于互联网应用的快速开发，又能在高并发、高性能的开发场景中如鱼得水。在 2015 年，Go 语言在服务端程序开发和 Web 开发领域大放光彩的同时也成功介入到了移动端开发领域。也正因为此，越来越多的创业公司（尤其是互联网应用和云计算领域的创业公司）选择 Go 语言作为其技术栈的重要组成部分。在国内，据不完全统计，已在生产环境中使用 Go 语言程序的知名互联网公司已有百度、美团、360、京东、搜狐、豌豆荚、宜信、微影时代 等等，更不用说国内日渐增多的云计算创业公司了。由此可见，对于广大的互联网软件开发者而言，关注和学习 Go 语言已经是已经很有必要的事情了。

回顾

2016 年已经到来，距 Google 在 2012 年 3 月发布的 Go 1.0 已将近 4 年。在 2015 年，Go 语言发生了不小的变化。从该年初发布的 1.4 版本到该年 8 月下旬发布的 1.5 版本，Go 语言终于完成了自举的过程，即：几乎完全用 Go 语言程序重写了自己，仅留有少许汇编程序。Go 语言的自举非常彻底，包括了最核心的编译器、链接器、运行时系统等。显然，这是一个很有意义的过程，代表着能力和自信。与此同时，Go 语言的运行时性能得到了大幅提升，尤其是在 1.5 版本完成的并发 GC 使得 Go 语言程序在响应时间方面有了质的飞跃。另外，Go 语言所支持的操作系统和计算架构越来越多，几乎涵盖了现今主流甚至非主流的所有选项。

当然，改变不止如此。下面，我们就列举几个比较惊艳的改变，并稍加剖析。

并发的 GC

GC，一般认为是 garbage collector 的缩写形式，通常被译为垃圾回收器。不过，它有时候也被看做是 garbage collection 的缩写形式，中文译为垃圾回收。在下文中，当 GC 被当做动词用时，指的是垃圾回收。但当 GC 被当做名词用时，指的是垃圾回收器。

在 1.4 以及之前版本的 Go 语言中，每次 GC 都会导致完全的“stop the world”（也可称之为 STW）。这意味着在 GC 期间，Go 语言的运行时系统会让调度器暂停对已启用的 Goroutine 的一切调度。也就是说，任何未处于运行状态的 Goroutine 都不会被递交至内核线程和运行，直到当次 GC 完成。如此暂停的代价不容忽视，对于有高并发需求的程序来说有时会显得非常棘手。在 1.5 版本出来之前，Go 语言的 GC 也常常因此被开发者们诟病。

Go 1.5 的 GC 是一个非分代、无转移的采用标记 – 清扫算法和三色标记法的并发垃圾回收器。它大刀阔斧地利用各种手段大大缩减了 STW 的时间。Go 语言官方保证，在 50 毫秒的 Go 程序运行时间中因 GC 导致的调度停顿至多只有 10

毫秒。有了这一保证，相当于为 Go 程序设定了一个响应时间的上限。对于对响应时间敏感的程序（许多互联网程序都是如此），这绝对是一个重大利好。当然，如此质的飞跃并不是一蹴而就的。实际上，Go 1.4 也为此做了很多铺垫，比如对 Goroutine 栈的改造以完全保证 GC 的标记操作的准确无误。图 1 展示了在最近几个版本的 Go 语言中 GC 的 STW 时间与内存堆大小的对应关系。

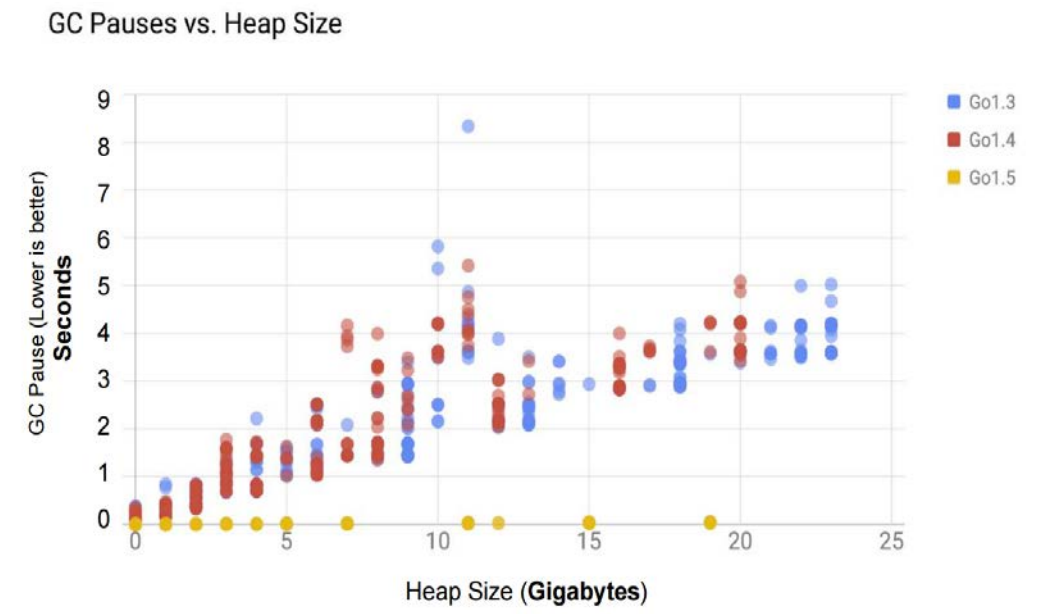


图 1 GC Pause vs. Heap Size

自带标准工具的更新

Go 语言的亮点之一就是自带了很多标准工具以帮助开发人员方便地进行 Go 程序的检查、格式化、编译、测试、部署，甚至升级。这些工具已经涵盖了一个软件的生命周期的方方面面，极大的方便了 Go 程序的开发者们。在 1.4 版本中，Go 语言的标准工具集中加入了 go generate。顾名思义，这是一个用于生成 Go 语言代码的命令。有意思的是，这源于一个几乎所有的计算机程序研发者们都有过的梦想——让计算机程序自己编写程序。go generate 命令可以利用 YACC（Yet Another Compiler Compiler，一种编译器的生成器）并根据某种描述文件来生成 Go 语言代码。不过，千万不要被这句话吓到。即使我们不懂 YACC，

甚至对 Go 语言的 AST（Abstract Syntax Tree，译作抽象语法树）一无所知，也可以使用 go generate 命令。比如，我们可以利用 go generate 命令把一些 HTML（Hypertext Markup Language，译作超文本标记语言）页面模板文件内置到生成的 Go 程序代码文件中（顺便说一句，Go 语言有自己的 HTML 页面模板语法，可用于编写 HTML 页面模板）。这样就无需在部署用于 Web 站点的 Go 程序时携带那些额外的文件了。下面展示一小段用于实现此功能的代码：

```
package main

//go:generate tpl_loader -ipath ../view -opath ../view
```

程序员们应该可以猜到最后一行代码实际上是一行注释。实际上，只要有了这行注释（其中的 tpl_loader 是笔者写的一个小工具，也非常的简单易懂），再在当前代码包目录下运行 go generate 命令，就可以实现上述功能了。在 Go 语言官方博客中可看到更详细的说明。

在 2015 年里，Go 语言在标准工具方面的增进还不止于此。一个更加令人兴奋的标准工具——go tool trace——随着 1.5 版本的发布而到来。Go 程序开发者们可以利用这一工具来图形化的展示出 Go 程序的追踪文件。当然，在展示之前，我们先要通过某种方式生成这样的文件。Go 语言为我们提供了三种用于生成 Go 程序追踪文件的方法：通过显式调用指定的标准库函数以手动生成、导入指定的标准库代码包以使其自动生成，以及在运行程序测试时添加指定标记来生成。此后，当我们运行 go tool trace 命令并把相应的可执行的 Go 程序文件和 Go 程序追踪文件作为参数以后，就可以在你的默认 Web 浏览器中看到类似下图的图形化展示了。

如图所示，Go 程序追踪文件可以呈现出对应 Go 程序的内存使用、Goroutine 和内核线程状态、调度过程等信息。为我们调试 Go 并发程序提供了非常有力的辅助。

图 2 Go 程序追踪文件的图形化展示

除了上述两个新增的标准工具之外，Go 语言官方也对一些已有的标准工具做了改进，比如：为 go build 命令和 go test 命令新增了可用标记以使其更加灵活、增强了 go tool vet 命令和 go doc 命令的功能，等等。

访问控制的增强

Go 语言对代码包的访问控制的进一步增强始于 1.4 版本。在 1.4 版本之前，Go 语言对程序实体（包括变量、常量、类型声明、函数等）采取的是两级访问控制策略。程序实体的名称的第一个字母的大小写决定了它的可访问范围。若为大写，则该程序实体可以被存在于任何位置的代码访问到。我们称这样的程序实体是公开的。若为小写，则该程序实体仅能被存在于当前代码包的代码访问到。我们称这样的程序实体是包级私有的。这样简明扼要的规则非常容易被记住。

到了 Go 1.4 时代，第三种访问控制规则出现。官方称之为 Internalpackages。其含义是，如果代码包 A 直接包含了一个名为 internal 的子代码包，那么这个 internal 包中公开的程序实体仅能被存在于代码包 A 及其子代码包中的代码访问到。这相当于使 Go 程序代码有了“模块级”的访问控制。

不过，在 Go 1.4 中，这个“模块级”的访问控制只是对 Go 语言源码和标准库中的（即 <Go 语言安装目录 >/src 中的）代码包有效，而对于其它代码包则是无效的。也就是说，这是一个过渡阶段。为的是让广大 Go 程序开发者有个适应期。到了 Go 1.5，由 internal 代码包代表的“模块级私有”访问控制规则才对所有代码包有了强制约束力。至此，Go 语言拥有了一套完整的三层访问控制策略。相比于之前的两层访问控制策略，它更加灵活和完善。

跨平台编译

在 1.5 版本之前，我们要想实现 Go 语言程序的跨平台编译是相当困难的。虽然因此催生出了几个开源的辅助工具，但其步骤也依然是相当繁琐的。其最主要的原因是那时的 Go 语言编译工具是由 C 语言编写的，是平台相关的。这里的平台相关，是指被编译后的程序的运行必要条件包含了目标计算机（也就是用于运行该程序的那台计算机）的操作系统和计算架构。其中，操作系统的可选项有 windows、linux、darwin 等，而计算架构的可选项目前有 386（即 32 位计算架构）、amd64（即 64 位计算架构）和 arm（一种基于精简指令集的计算架构，多用于便携设备专用 CPU）。例如，我们在 32 位的 Windows 操作系统下，使用平台相关的 Go 语言编译工具编译的程序是不能在 64 位的 Linux 操作系统下运行的，反之亦然。

我们已经知道，Go 1.5 的所有编译工具和运行时系统都被用 Go 语言重写了。这使得这些编译工具被进行了一系列合并，并且跨越了平台的界线。例如，之前针对不同计算架构的 Go 语言编辑器 5g、6g 和 8g 被合而为一并命名为 compile。随之而来的优势就是，我们可以使用这些编译工具轻而易举的进行跨平台的程序编译操作。我们只需在程序编译操作之前设置一下目标计算的操作系统和计算架构。前者通过设置环境变量 GOOS 来实现，而后者通过设置环境变量 GOARCH 来实现。例如，当我在 64 位的 Linux 操作系统下通过执行如下命令来编译一个 Go 源码文件之后，就可以在 32 位的 Windows 操作系统下直接运行那个编译后的结果文件了。

```
env GOOS=windows GOARCH=386 go build main.go
```

对 Android 和 iOS 开发的支持

从 Go 1.4 开始，开发人员已经可以利用官方扩展库中的 mobile 代码包来编写 Android App 了。到了 Go 1.5，其对 iOS App 的支持也已可用。这无疑是一大里程碑，使人振奋！它使得 Go 语言的适用领域大大拓展，并垒砌了其在移动互联网时代甚至物联网时代的根基。开发者们既可以使用 Go 语言来构建原生的 Android App 或 iOS App，也可以用它来编写可被 App 程序调用的基础库。读者可以通过官方文档来了解相关步骤，也可以去看看 Go 语言的创始人之一 Rob Pike 为大家编写的示例 App。另外，大家也可以关注 Go 语言北京用户组和北京 GDG（Google Developer Group - Beijing）联合主办的 Go 语言技术聚会，其中会包括与此有关的 Topic。此聚会的时间初定在 2016 年 2 月的下旬。

其他调整

除上述比较突出的变化之外，Go 语言在很多地方也做了调整。比如，Goroutine 内存栈的增长方式的变更，Goroutine 内存栈的初始大小由 8K 缩减为了 2K、GOMAXPROCS 的默认值由 1 变成与当前计算机的 CPU 核心数一致、Go 代码可以被用于生成动态链接库了，等等。对于这些调整，笔者就不一一细说了。不过，它们对于 Go 语言在 2015 年的精进也都起到了一定的推动作用。

总之，Go 语言在 2015 年的发展迅速且振奋人心。无论在其本身的功能、性能和适用领域上，还是在社区方面（尤其是在中国）都是如此。如果说笔者在著《Go 并发编程实战》这本书的时候还只是建议大家把 Go 语言作为自己的第一或第二编程语言并以此作为长线技术投资的话，那么现在我强烈建议所有互联网软件开发者都去尝试并使用 Go 语言构建他们的（个人或公司的）软件系统，并真正将其作为手边的常用工具。

Go 语言号称是云计算时代的 C 语言。它也正在持续、快速地向着这一目标前进。如果你也打算跟上后云计算时代、物联网时代以及不久就会出现的人工智能时代的话，那么就很有必要玩儿转 Go 语言了。我相信，它一定不会让你失望。

展望

在 2016 年，我们目前可见的 Go 语言进展就是预计在 2 月份发布的 1.6 版本了。在这个版本中，Go 语言官方准备重点发展 UI 库。这也是 Go 语言当前的一大短板。在官方的 UI 库完全就绪之前，笔者希望大家去关注一下七叶（优秀的国产 Go 语言 IDE——LiteIDE——的作者）的新项目 GoQt。此项目就是一个基于 QT 的 Go 语言 UI 库。鉴于 LiteIDE 的优秀，我非常看好这个项目。另外，Go 语言在程序测试支持、程序运行分析以及程序调试方面都会有所改进，尤其是后者。实际上，许多不适应使用 Go 语言开发程序的程序员的最大抱怨就是 Go 语言程序不易调试（不过大家可以去了解下真正的 Go 语言爱好者是怎样调试 Go 程序的）。Go 语言官方也在 2015 年的重大版本升级中对这一方面做出了很多改进。随之而来的就是 Go 语言在于各大主流 IDE 的集成方面所作出的不断努力。在 Go 1.6，这种努力还会继续。程序的开发效率与运行效率同样重要。甚至在某些时候，前者比后者更加重要。这也是许多脚本语言得以生存并繁荣发展的重要原因之一。Go 语言的创造者们更是深谙此道。最后，Go 语言还会在移动 App 开发方面进行一步的增强。笔者相信 Go 语言在这一开发领域一定会有长足的进步的。

在奇点临近的当下，由于各种智能移动设备的先天优势，移动开发需求持续暴涨。我想，大多数互联网软件开发团队（尤其是创业团队）都会想用尽量精简的技术栈去支持多方产品的需求。如果能用一种既可快速上手又能高效运维的技术那该有多好。如果你通读本篇并有所思考，就很可能有与我相同的感受——Go 语言就是这样的一门技术。如果你觉得理由并不充分或想深入了解 Go 语言，那么就加入到Go语言社区并切实的感受一下吧。笔者发起的Go语言北京用户组（微信公众号：golang-beijing）也欢迎你的加入。

在不久的将来，Go 语言一定会实现在程序开发领域的全面覆盖。到那时，Go 程序员的含金量也就毋庸置疑了。那么，我们为什么不现在就去做如此重要的技术投资或去积极的在内部培养 Go 开发工程师呢？

作者简介：郝林，Go 语言北京用户组的发起人，著有图灵原创图书《Go 并发编程实战》，同时也是在线免费教程《Go 命令教程》和《Go 语言第一课》的作者。现在微赛时代担任平台研发负责人。

解读 iOS：惊喜和机遇

作者 徐川

2015 年过去了，对于 iOS 开发者来说，这是充满惊喜和机遇的一年。Swift 开源让这门语言有了更光明的前途，并且让苹果和开发者之间的联系更紧密了，React Native 给 iOS 开发带来了全新的理念，watchOS 和 tvOS 则给开发者开辟了更广阔的市场。下面就让我们来具体回顾一下。

iOS 9、watchOS 2、tvOS

苹果在 2015 年发布了 iPhone 6/6s、iPad Pro 等新设备，一如既往的高品质，值得信赖，在市场方面，苹果也取得了佳绩，其大中华区的销售业绩和市场份额都有可喜的进步，其地位已经无法被轻易撼动。

不过作为开发者，更关注的还是 WWDC 苹果开发者大会，在本届大会上，库克发布了 iOS 的最新版本 iOS 9 及相应的 SDK，其更新包括 Multitasking、App Slicing、App Linking 等，开放了更多能力和新特性给开发者，而用户对其接受度也更高，发布不到两个月装机率达到 66%，成为主流的系统版本。

除了 iOS 这个已经相对成熟的平台，苹果在可穿戴设备和智能家庭方面也逐渐开始发力，其中的代表就是 watchOS 和 tvOS。watchOS 2 作为第一代产品的软件升级，它稍微放开了一些限制，新系统中 Watch App 的 extension 将不像现在这样存在于 iPhone 中，而是会直接安装到手表里去，Apple Watch 从一个单纯的界面显示器进化为了可执行开发者代码的设备。tvOS 则是苹果在发布新版 Apple TV 时的系统升级，相对于之前的保守，tvOS 引入了应用商店，可以运行第三方应用，这让电视成为一个新的平台，截止到 2015 年早期 Apple TV 销量已经超过 2500 万台，对于开发者来说这已经是不可忽视的市场。

当然，watchOS 和 tvOS 这两个平台仍然处于相当早期的阶段，相应设备的市场也并没有完全打开，苹果对它们的开放很谨慎，开发者目前能做的事情还比较有限，但它们都有可能复制 iOS 的成功，因此它们的潜力不容忽视。

Swift 开源

对于 iOS 开发者来说，2015 年可以说是属于 Swift 的一年，在 WWDC 现场，库克宣布 Swift 开源引起了全场欢呼，在正式开源后更是成为 Github 有史以来关注（star）增长最快的项目。

编程语言是有信仰的，相对于已有数十年历史的 Objective-C 来说，Swift 的语法更加现代，融合了多种语言中的优点，获得了不少拥趸。在 Swift 开源之前，就有人尝试在 Android 上运行 Swift 编写的应用，还有人开发了 Swift 服务端开发工具包，在 Github 上，基于 Swift 编写的开源项目也呈快速上升趋势。

这次开源对苹果同样也是意义重大，苹果与开发者之间的距离从未这样近过。在以前，苹果虽有 Webkit、LLVM 等开源项目，但领域相对垂直，参与项目难度较大。现在苹果把 Swift 及其相关的项目放到了 Github 上，一般的用户也可以很轻松的关注项目的进展、与开发人员沟通，甚至是给项目做贡献。而从苹果到现在的举措来看，这个项目并不仅是代码托管，而是彻底的社区化运作，开放程度前所未有的大。

与 Swift 相关的项目，苹果也以开源社区的形式运作，如 Swift 的包管理项目，吸收了社区的传奇人物 Max Howell、Mattt Thompson 等，而官方也鼓励开源社区翻译 Swift 的官方文档，像梁杰组织翻译的中文版文档即被 Swift 官网推荐。

可以预见，2016 年，开源的 Swift 将更加的强大，将于 2016 年夏季发布的 3.0 版本的目标规划已经发布在 Github 上，只要你愿意，你就可以参与到 Swift 的未来中。

React Native

2015 年对 iOS 开发影响重大的事情还有一个，那就是 React Native 框架，它分别在 3 月份和 9 月份发布了 iOS 和 Android 的开源版本，每一次都引起大量的关注和讨论。

React Native 最重要的特性是跨平台，即 Facebook 所宣传的“Learn Once, Write Everywhere”，还有由于引入前端开发技术带来的更新和 Hotfix 上的便利，超过以往基于 Lua 的方案，还催生出 AppHub 这样的开发服务。

由于 React Native for iOS 版发布较早，早有人用它进行了实验性开发，但用于正式应用开发的并不多。由于项目还处于早期，不可避免会出现一些坑，相关的第三方库和学习资源也十分匮乏。但对于 React Native 来说，这只是时间的问题。

React Native 并不是孤身作战，它的根源在 React。2015 年同样是 React 的风云之年，这一年 Facebook 接连发布了 React Native、Nuclide、GraphQL、Relay，以及社区各种 Flux 实现，一个完整的 React 生态圈已见雏形，一旦成熟起来，不止是移动开发，未来的大前端开发都将被彻底改变。

目前，能够验证 React Native 能力的正式应用很少，相信到 2016 年，基于 React Native 的典型应用和开发方案将会出现，将这个移动开发解决方案推到一个新的高潮。

社区与开源

2015 年 iOS 开发社区也在蓬勃发展，特别是国人在这方面取得了非常大的进步，不但贡献了优秀的开源项目，还第一次由社区主办了 Swift 开发者大会，获得了社区的积极响应。

至于开源，则几乎成为 iOS 开发者展示技术能力、学习和交流的身份标识，除了 Swift 和 React Native 之外，笔者挑选了一些 2015 年值得关注的开源项目如下：

1. **RxSwift 和 ReactiveCocoa**：2015 年函数响应式编程火遍了移动开发领域，ReactiveCocoa 相对老资格一些，在 2015 年发布的 3.0 版本支持了 Swift 接口，RxSwift 则是 ReactiveX 的 Swift 版本，功能更加强大，有后来居上的趋势。
2. **JSPatch**：由国人开发的 iOS 应用 Hotfix 库，由于它小巧灵活、功能强大，现在已被各种商业应用所广泛使用，大大方便了 iOS 应用紧急问题的修复。
3. **YYKit**：同样是国人开发的开源库，包括一系列的工具库，极其强大，同时其源码也是学习的好资源，唐巧对它的作者进行了专访，了解了它背后的故事。

2015 年涌现的优秀 iOS 开源项目当然不止这些，由于篇幅所限这里就不一一列举了。希望这些开源项目能激励更多的国人 iOS 开发者参与到开源中来。

小结

iOS 9 带来了新特性、watchOS 和 tvOS 带来了新市场，Swift 刚刚开源，无论是直接参与贡献，还是基于 Swift 做工具、分享知识都非常必要，React Native 也缺乏成功案例和成熟的技术方案，需要社区去完善和实现。面对这种种机遇，需要善于学习、勇于开拓创新的开发者去努力抓住，相信 2016 年对于 iOS 开发来说，将是更加精彩纷呈的一年。

解读私有云：政企私有云市场风云录

作者 周静

笔者参加 AWS re:Invent 2015 时感受颇深的一点正是国内外云计算行业在市场方面迥异。恰好近日又收到品高软件副总经理周静女士的这篇《2015 年政企私有云市场风云录》，是以之为云计算解读 2015 之私有云篇。我大胆地用美团网创始人王兴在饭否上说的这句“2014 年，该上市的上市；2015 年，该合并的合并；2016 年，该倒掉的倒掉。”来形容一下国内的私有云市场发展，缪误之处还请各位方家批评指正。

2015 年的政企私有云市场

私有云市场其实是传统的政企市场。这个领域原本都是经过外企教育多年市场——IBM、微软、Oracle 等大企业都属于这一阵营——被规划好了基础架构、应用架构甚至 IT 管理架构，并培训了政企客户的 IT 管理员。因此，要认识企业云，必须认识到，云的构建是这一切重塑和调整的过程。

2014 年以来，国内外政企私有云市场渐渐起来。客户开始谈虚拟化说“我也需要云”，开始有了大量的考察、调研，同时也普遍焦虑。这个焦虑的原因不是因为云，而是互联网给自身业务带来的巨大冲击和挑战。IT 只是支撑业务的，在这个迷茫期，企业都在想但很少动。2015 年被行业称为 to B 的元年，政府、企业对 IT 的建设欲望空前强烈，随着政府“互联网 +”的政策导向以及实体经济面临互联网经济的挑战加剧，大家都在寻找适合自己的 IT 方案，目前这些方案无一例外都指向建立在云的基础之上。在这个过程中，中国少有的跟全球站在近乎同一时间线上。

对标美国政企市场

美国作为世界 IT 毋庸置疑的领头羊，过去完全是全球 IT 技术的发源地和示范者。分析这个市场，无疑能够对我们有所启迪。

亚马逊一家独大，互联网逆袭，政企市场变革在即

亚马逊是从不接受有私有云的，从云出现的那一天，他就认为只有公有云不会再有私有云。唯有基础设施的集中管理，才能真正发挥云的经济性、弹性、可靠。因此，即使亚马逊开放了 VDC（Virtual Data Center）的接口，让企业内部网络接入管理，但也从不提混合云。唯一的例外是 2013 年与 IBM 争夺美国中情局 6 亿美元的大单，为 CIA 搭建了专有云，此后再无案例。

对公有云的专注使得亚马逊在自己的发展道路上越走越远。从数据库到 IOT（物联网），亚马逊在不断侵蚀原有 IT 厂商的赚钱金矿。而 IT 的基础设施，也在被亚马逊重新定义。随着能力差距的不断拉大，其它厂商在基础设施上与亚马逊争夺市场的能力越来越弱。纵观各大厂商 2015 年发布的财报不难得知，公有云基础设施的竞争已经结束。

毫无疑问，基础设施、数据库、数据分析对政企用户的吸引力是巨大的，但是由于原有系统的架构问题，过去的政企客户还缺乏产业链的协同和服务。这是亚马逊唯一留给其它 IT 厂商的机会。

微软、IBM 加速变革，机会与挑战并存

真正提出用户私有云的是惠普、微软和 IBM。已有的客户基础是其对抗已经成熟且凶悍的亚马逊的唯一法宝。而私有云的价值，短期之内也不可能被忽略。随着基础设施云化过程中标准化程度的逐步加强，基础设施运营能力的差异正在逐步缩小，在业务和行业上的竞争，将会是另一个重新洗牌的机会。

因此，2015 年最被寄予厚望的厂商是微软，“移动优先、云优先”作为战略被提上日程。微软开启移动端与 PC 端联动，移动端支持苹果、支持安卓，拥抱开源的产品策略加上已有的巨大用户群和使用习惯以及最完整的产品线，在政企市场优势极大。只要微软不犯错、不过分贪婪，未来一定是亚马逊劲敌。

IBM 在金融、医疗等大型行业拥有多年的积累，具备比其它企业都要丰富的行业知识和能力。Bluemix 的推出，显然是蓝色巨人将在 PaaS 层面用自己在人工智能的积累筑为护城河。联手苹果的战略，同样出于云端一体化的策略。能否整体翻身尚不好说，但提供某些行业内一体化的服务，还是很有机会。

近观中国政企市场

中国市场由于缺乏公正权威的评估和成熟的信息披露机制，一直处在云山雾罩之中难以看清。如果从对标美国企业来看看中国厂商的策略或许有点收获。

阿里云能否能在政企市场保持优势

中国的互联网市场已是 BAT 的天下。中国的市场特点，几乎决定了 to C 的市场短期内不会再有逆袭的可能。阿里云从开始就把自己定位成中国的亚马逊，从 2011 年发展至今，无论是功能、规模，在中国的公有云市场已很难有竞争者。然而，由于中国的个人创业者基本属于从 BBS 站长之类发展而起，与美国的创业公司在规模和能力有一定差距。因此，中国的公有云市场，主要还集中在稳定运行保障上，在产品和服务上还有一段路要走。

阿里与亚马逊不同的是，中国的 IT 成熟度较低，IT 消费市场集中在大中型政企。为了更多的市场占有和营收，阿里必须开始放弃单一公用云战略而走向政企云。这个战略无疑将面对产品研发、运维服务、人才结构等方面的巨大挑战。阿里已经在各地建政务云、为海关建私有的大数据云、为公安部建设公安大数据云，于 2015 年占领了不少的政企市场。不过政企市场需要对业务的理解和产业合作，这会跟亚马逊走一条不一样的路，需要时间检验战略的可行性。

华为、华三、品高、青云各走什么路

华为、华三是国内较早走 OpenStack 路线的龙头企业，尤其是华为，放弃了自研两年多的私有云产品，全面转向 OpenStack 路线。华为和华三都是从网络到服务器到存储的全设备生产厂商，云有助于打破硬件产品的同质化竞争，具备更高的黏度。由于硬件厂商的销售渠道已经成熟，软硬一体化有更强的价格竞争能力。

2015 年私有云市场启动后，华为、华三是真正的获利者。不过，OpenStack 虽然得到很高的赞誉，但毕竟还正在发展中，其成熟性和服务生态都有待考验。华为、华三在软件服务领域的能力自然不是强项，借力 OpenStack 生态来销售自家产品这条路能否像硬件标准化一样也是个问题。值得注意的是，华为在 2015 年 7 月 30 日宣布企业云战略，提出聚焦垂直行业、发力混合云。面对外界对华为走出幕后与运营商客户正面竞争的质疑，提出可以与运营商共同发展业务。同时，华为也一直保持着与 Vmware、Ctrix 等国际厂商亦合作亦竞争的态势，也基本表明了华为在企业级云市场上，是一个聪明的既得利益者，并不绑定在一种路线上。

私有云市场上完全以软件为主体产品服务销售策略的值得一提的是品高云和青云。品高云从 2015 年一月 Forestor 的国内私有云象限图进入大众的视野，但早在 2008 年品高软件就开始研发云计算基础架构平台，两年后国内第一套商用基础云架构产品 BingoCloud 1.0 版正式亮相，甚至比 OpenStack 的发布还早。从 2010 年开始以每年一个版本的速度迭代，2015 年 10 月 21 日，BingoCloud 6.0 发布。企业级开发服务起家的品高软件，拥有丰富的企业信息系统经验和

超过 600 人的技术服务队伍，已形成不靠资本输血的现金流业务，面对各种业务机会，能更从容地选择和服务，这是初创型企业难以企及的。在 BingoCloud 6.0 发布会上，品高云一反低调的常态，宣布自己是国内政企基础架构云的唯一商业选择。并 Show 出一堆有实力的成功案例，国内规模最大的私有云腾讯私有云、国内最多电子政务云的成功案例、融合高性能计算的教育云、可纳管替换 VMware 的金融云、公私混合的托管方案等等，另外在诸多实力企业的技术评测结果也显示了这个低调的南方企业的技术实力。

青云从诞生就一直是资本市场的宠儿，创始人的 IBM 研发背景和在公有云市场以其秒起的特性塑造成技术型公司的典型。自 2014 年开始，青云开始进入政企云市场，并发力金融云。与品高云一样，青云不采用 OpenStack 的方案而是自主研发的技术架构，因此，既有产品更可控、更稳定可靠的优势，又面临企业对其技术路线的质疑和服务能力的担心。而青云作为初创公司的人员规模，想要在企业级市场立足，生态系统的构建显然是唯一出路。与品高在 2014 年开始生态建设的初衷一样，这也是青云在 2015 年重金建设渠道生态的原因。但国内企业能够建设好生态环境的案例显然不多，让我们拭目以待吧。

OpenStack 生态圈的产品和服务重于技术

在 2015 年，OpenStack 的发展可谓喜忧参半，作为大型公有云未来的承载平台，似乎机会渺茫。创始者之一的 Nebula 关闭、HP 的退出、创始公司 RackSpace 的困境都似乎佐证了这一点。私有云于是成了 OpenStack 的主要战场，第三大代码贡献者 Mirantis 获 1 亿美金 C 轮融资正是发生在 2015 年。

认真分析 OpenStack 的客户与市场不难发现，大型的集团企业或行业的领导者，希望利用开源技术实现对 IT 资源更高的把控力。而成功使用的企业，无不显示出自身较强的 IT 研发和运维能力。而暂时成功的 Mirantis，将自己定位于众多大型企业的 OpenStack 研发合作伙伴和产品经理。可以说是除了像 RedHat 那样提供发行版外，开源软件的一条新型发展之路。

OpenStack 公司，他们在哪里

2015 年 OpenStack 更加火热，在中国市场，UnitedStack 有云强调自己找到了做企业托管云这一合适的市场，并获得了思科的投资依然站在 OpenStack 创业公司的第一战线。EasyStack 以其简单好用，提出企业不需要那么复杂的私有云管理好虚拟化就足够，也在社区有一定知名度。而年度最重大事件无疑是国际上风生水起的 Mirantis 与 UCloud 成立合资公司，提供私有云 / 混合云解决方案。当然现在进入中国市场能否在服务能力和本地化上复制成功还需要时间检验。但是 Mirantis 在国外大企业成功部署 OpenStack 的服务能力，显然备受国内希望选择 OpenStack 技术企业的期待。

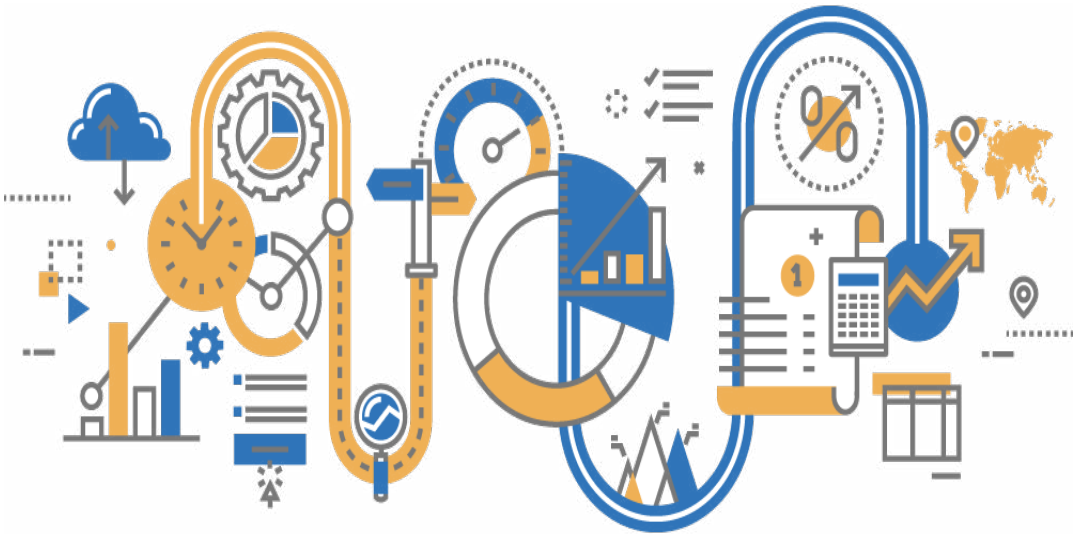
国内投入 OpenStack 的创业型、服务型企业不少，能否在 2016 年真正获得更多用户，找到自身的定位（做 OpenStack 的发行版厂商还是服务商），将是未来几年内成为中国的 Nebula 还是 Mianctis 的关键。

政企云 2016 年市场展望

政企云市场在 2014 年开始升温，2015 年可以说正式启动，2016 年的市场很值得期待。但是云的市场是否就是 IaaS 的市场？企业一旦开始上云，就意味着未来整个技术架构、管理模式的变迁。从顾问服务到架构迁移，再到目前最热的移动化架构、大数据服务、应用的 PaaS 服务、解决应用开发管理问题的容器架构，都会是企业的思考点。最终会像亚马逊所说企业使用公有云还是像 Gartner 所说企业需要混合云？这些都会在不断的尝试中演进。2016 年这个市场必将爆发，但是竞争只会在已经全方位做好准备、有帮助企业入云经验丰富的厂商中产生。

解读 Spark：新生态系统的形成

作者 梁堰波



2015 年的 Spark 社区的进展实在是太快了，我发现 1 月份出版的一本参考书到现在已经有很多内容是过时的了。社区大踏步前行的同时，用户和应用案例也越来越多，应用行业越来越广泛。到年底了我们来梳理下 Spark 这快速发展的一年。

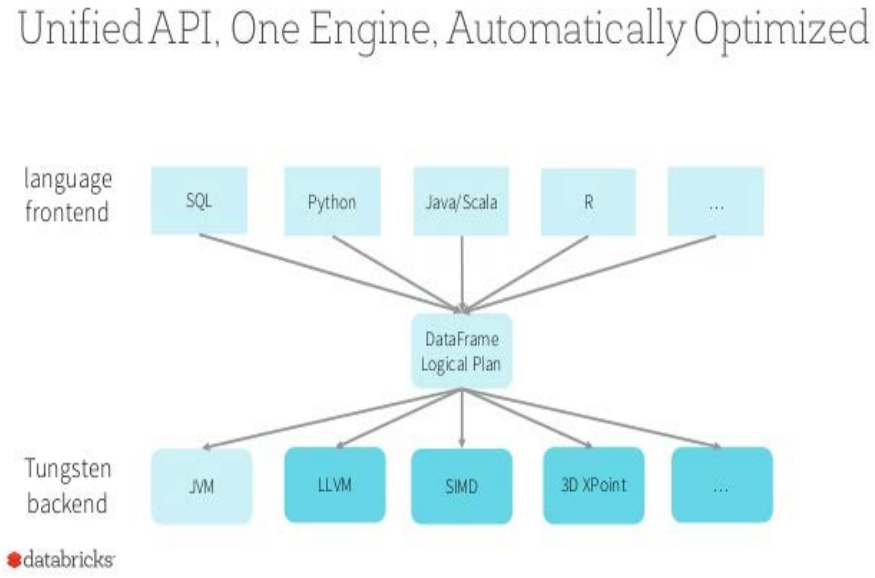
先从全局有个认识，我尝试用三句话来概括下 Spark 最主要的变化，然后在接下来的篇幅选取一些重点内容展开。

1. Spark 生态系统渐趋完善。支持的外部数据源越来越多，支持的算子越来越丰富，自身的机器学习算法越来越完善。同时在 API 支持上也有很大进步，新增加的 R 语言 API 使得 Spark 能被更多的行业所接受。
2. Spark 的应用范围和规模在不断扩大。在互联网和电子商务行业的应用不断增多、规模不断扩大的基础上，越来越多的金融、电信、制造业等传统行业也开始使用 Spark 解决他们遇到的大数据问题。

3. Spark 自身的性能和稳定性在不断提升。Tungsten 项目让 Spark 跑的越来越快，越来越多的代码贡献者和使用经验让 Spark 越来越稳定。相信明年发布的 Spark 2.0 将是一个里程碑式的版本。

转向以 DataFrame 为核心

在传统意义上 Spark 的核心是 RDD 和 RDD 之上的各种 transformation 和 action，也就是各种算子，RDD 可以认为是分布式的 Java 对象的集合。2013 年推出了 DataFrame，可以看做分布式的 Row 对象的集合。DataFrame 除了提供了比 RDD 更丰富的算子以外，更重要的特点就是有执行计划的优化器，这样用户只需要指定自己的操作逻辑，DataFrame 的优化器会帮助用户选择一条效率最优的执行路径。同时 Tungsten 优化（下一章重点讲）使得 DataFrame 的存储和计算效率比 RDD 高很多。Spark 的机器学习项目 MLlib 的 ML pipeline 就是完全基于 DataFrame 的，而且未来 Streaming 也会以 DataFrame 为核心。



Tungsten 让 Spark 越来越快

那么为什么 DataFrame 比 RDD 在存储和计算上的效率更高呢？这主要得益于 Tungsten 项目。Tungsten 做的优化概括起来说就是由 Spark 自己来管理内存而

不是使用 JVM，这样可以避免 JVM GC 带来的性能损失；内存中的 Java 对象被存储成 Spark 自己的二进制格式，更加紧凑，节省内存空间，而且能更好的估计数据量大小和内存使用情况；计算直接发生在二进制格式上，省去了序列化和反序列化时间。

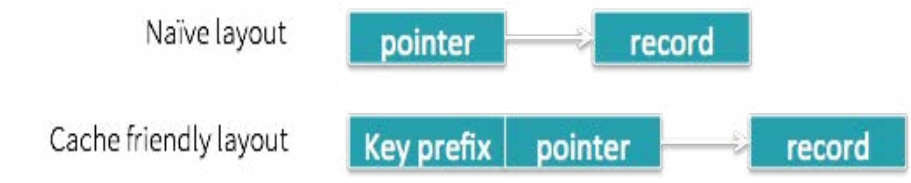
像传统的 Hadoop/Hive 系统，磁盘 IO 是一个很大的瓶颈。而对于像 Spark 这样的计算框架，主要的瓶颈在于 CPU 和内存。下面看看 Tungsten 主要做了哪些优化：

1. 基于 JVM 的语言带来的问题：GC 问题和 Java 对象的内存开销。例如一个字符串”abcd”理论上只有 4 个 bytes，但是用 Java String 类型来存储却需要 48 个 bytes。Spark 的改进就是自己管理内存，不用 JVM 来管理了，使用的工具是 sun.misc.Unsafe。DataFrame 的每一行就是一个 UnsafeRow，这块内存存的啥东西只有 Spark 自己能读懂。有了这种特有的二进制存储格式后，DataFrame 的算子直接操控二进制数据，同时又省去了很多序列化和反序列化的开销。

2. Cache-aware 的计算。现在 Spark 已经是内存计算引擎了，但是能不能更进一步呢，能不能更好的利用 CPU 的 L1/L2/L3 缓存的优势呢，因为 CPU 缓存的访问效率更高。这个优化点也不是意淫出来的，是在 profile 了很多 Spark 应用之后得到的结论，发现很多 CPU 的时间浪费在等待从内存中取数据的过程。所以在 Tungsten 中就设计和实现了一系列的 cache-friendly 的算法和数据结构来加速这个过程，例如 aggregations, joins 和 shuffle 操作中进行快速排序和 hash 操作。

以 sort 为例，Spark 已经实现了 cache-aware 的 sort 算法，比原来的性能提升至少有 3 倍。在传统的排序中是通过指针来索引数据的，但是缺点就是 CPU cache 命中率不够高，因为我们需要随机访问 record 做比较。实际上 quicksort 算法是能够非常好的利用 cache 的，主要是我们的 record 不是连续存储的。Spark 的优化就是存储一个 key prefix 和指针在一起，那么就可以通过比较 key prefix 来直接实现排序，这样 CPU cache 的命中率就会高很多。例如如果我们需要排序的列是一个 string 类型，那么我们可以拿这个 string 的 UTF-8 编码的前 8 个字节来做 key prefix，并进行排序。关于这个优化可以

参见 SPARK-9457 和 org.apache.spark.shuffle.sort 下面的类，最重要的是 ShuffleExternalSorter 和 ShuffleInMemorySorter 两个类。



3. 运行时代码生成：运行时代码生成能免去昂贵的虚函数调用，同时也省去了对 Java 基本类型装箱之类的操作了。Spark SQL 将运行时代码生成用于表达式的求值，效果显著。

除了这些优化，我认为还有两个很重要的变化。

1.Unified Memory Management

在以前 Spark 的内存显式的被分为三部分：execution，storage 和其他。execution 内存用于 shuffle, join, sort 和 aggregation 等操作，而 storage 内存主要用于 cache 数据。在 1.6 版本之前是通过 spark.shuffle.memoryFraction 和 spark.storage.memoryFraction 两个参数来配置用于 execution 和 storage 的内存份额。从 1.6 开始这两部分内存合在一起统一管理了，也就是说如果现在没有 execution 的需要，那么所有的内存都可以给 storage 用，反过来也是一样的。同时 execution 可以 evict storage 的部分内存，但是反过来不行。在新的内存管理框架上使用两个参数来控制 spark.memory.fraction 和 spark.memory.storageFraction。

2.Adaptive query execution

这个特性说大了就是所有数据库最核心的一个功能 query execution optimization，可以做的东西非常多。我们自己写 Spark 程序中经常会碰到一个 job 跑到最后每个分区的数据量很小的情况，这是因为以前的 Spark 不会估

计下游 RDD 的每个分区的数据量大小，并根据数据量大小来调整分区个数。以前遇到这种问题就需要手工 repartition，用户自己要心里有数到哪个阶段的 RDD 的 partition 数据变多了还是变少了，需要跟着调整分区的数目，非常不灵活。从 1.6 版本开始有了部分支持，主要是能够估计在 join 和 aggregate 操作中 Shuffle 之后的分区的数目，动态调整下游 task 的数目，从而提高执行效率。

DataFrame 和 SQL API

Spark 从 API 的角度看，可以分为两大类：

- 类似于 Python 的 Pandas 和 R 语言的 DataFrame API，用户可以使用 Scala/Java/Python/R 四种语言调用这个 API 处理数据；
- SQL 语言 API。又分为两种：一个是普通的 Spark SQL，一种是 Hive SQL。

虽然 API 不同，但是背后解析出来的算子是一样的，DataFrame 的各种算子其实就是各种 SQL 的语法。Spark 在 SQL 语法的支持越来越丰富的同时内置的 SQL 函数得到了很大的增强，目前已经有超过 100 个这样的常用函数 (string, math, date, time, type conversion, condition)，可以说最常见的 SQL 内置函数都有了。

作为一个类 SQL 的分析工具，聚合函数是非常核心的。Spark 1.5 和 1.6 在聚合函数上都有很大改进：实现了一个新的聚合函数接口，支持了一些 build-in 的聚合函数（例如 max/min/count/sum/avg/first/corr/stddev/variance/skewness/kurtosis 以及一些窗口函数等），同时基于新接口实现了相应的 UDAF 接口。新的聚合函数接口是 AggregateFunction，有两种具体的实现：ImperativeAggregate 和 DeclarativeAggregate。ImperativeAggregate 类型的聚合操作就是通过用户定义三个动作 initialize/update/merge 的逻辑来实现聚合的；而 DeclarativeAggregate 则是通过指定 initialValues/updateExpressions/mergeExpressions 这三个表达式然后通过代码生成的方式来做聚合的操作。这两种方式各有利弊，一般来说代码生成效率更高，但是像

variance/stddev/skewness/kurtosis 这样的多个表达式需要依赖同一个中间表达式的场景下，代码生成的执行路径由于不能共享中间的结果，从而导致其不如 ImperativeAggregate 效率更高，所以在 Spark 内部的实现中这几个聚合函数也是通过 ImperativeAggregate 来实现的。

SQL API 上另一个变化是可以直接在文件上进行 SQL 操作，不需要把这个文件注册成一个 table。例如支持”select a, b from json.`path/to/json/files`”这样的语法，这个应该是从 Apache Drill 借鉴过来的。

另外一个里程碑式的特性就是 Dataset API(SPARK-9999)。Dataset 可以认为是 DataFrame 的一个特例，主要区别是 Dataset 每一个 record 存储的是一个强类型值而不是一个 Row。这个强类型的值是以编码的二进制形式被存储的，这种存储格式可以不用反序列化就直接可以被上面的算子(例如 sort,Shuffle 等)操作。所以在创建 Dataset 的时候需要指定用于这个编码工作的 Encoder。

这样一些需要强类型的地方就可以使用 Dataset API，不失 DataFrame 的那些优点，同时又可以帮我们做类型检查。所以从某种角度上说这个 Dataset API 在将来是要替换掉 RDD 的。

外部数据源和 Hive 支持

这个 feature 可以说是建立起 Spark 生态系统的基础，使得 Spark 与大数据生态圈的其他组件联系起来了。可以这么理解，你无论数据是在 HDFS 上，还是在 Cassandra 里面，抑或关系型数据库里面，我 Spark 都可以拿过来做分析和处理，或者机器学习，我这边处理完了你让我写到哪去我就可以写出去。这个特性使得 Spark 成为了大数据处理的核心一环。目前 Spark 支持的外部数据源有很多种，主流的像 Parquet，JSON，JDBC，ORC，AVRO，HBase，Cassandra，AWS S3，AWS Redshift 等。

在这些外部数据源中，Parquet 是其中最核心的，Spark 的 Parquet 支持也有了很大的改进：修复了越来越多的 bug，Parquet 的版本升级到 1.7；更快的

metadata discovery 和 schema merging；能够读取其他工具或者库生成的非标准的 parquet 文件；以及更快更鲁棒的动态分区插入；对于 flat schema 的 Parquet 格式的数据的读性能提升了大约 1 倍（SPARK-11787）。

另外在 Hive 支持方面，越来越多的 Hive 特有的 SQL 语法被加入到 Spark 中，例如 DISTRIBUTE BY... SORT 等。支持连接 Hive 1.2 版本的 metastore，同时支持 metastore partition pruning（通过 spark.sql.hive.metastorePartitionPruning=true 开启，默认为 false）。因为很多公司的 Hive 集群都升级到了 1.2 以上，那么这个改进对于需要访问 Hive 元数据的 Spark 集群来说非常重要。

机器学习算法

Spark 在机器学习方面的发展很快，目前已经支持了主流的统计和机器学习算法。虽然和单机的机器学习库相比 MLlib 还有一定的差距；但是纵观所有基于分布式架构的开源机器学习库，MLlib 是我认为的计算效率最高的。表 1 列出了目前 MLlib 支持的主要的机器学习算法。

下面简单说下其中一些亮点：

1.MLlib 已经有了对 Generialized Linear Model(GLM) 的初步支持：GLM 是统计学里一系列应用非常广泛的模型，指定不同的 family 可以得到不同的模型。例如“Gaussian” family 相当于 LinearRegression 模型，“Bionomial” family 相当于 LogisticRegression 模型，“Poisson” family 相当于 SurvivalRegression 模型。目前 MLlib 已经提供了这三种模型的机器学习解法和 LinearRegression 的 normal equation 解法。

下面详细说说这两种解法：一种是利用 WeightedLeastSquares(WLS) 优化方法；另一种是利用 L-BFGS 优化方法。前者是通过解 normal equation 的思路求解，只需要对所有的数据过一遍（不需要迭代）即可得到最后的模型（好像很神奇），同时可以算出像 coefficients standard errors, p-value, t-value 等统计指

	Discrete	Continous
Supervised	Classification LogisticRegression (with Elastic-Net) SVM DecisionTree RandomForest GBT NaiveBayes MultilayerPerceptron OneVsRest	Regression LinearRegression(with Elastic-Net) DecisionTree RandomForest GBT AFTSurvivalRegression IsotonicRegression
Unsupervised	Clustering KMeans GaussianMixture LDA PowerIteration Clustering BisectingKMeans	Dimensionality Reduction, matrix factorization PCA SVD ALS WLS

表 1

标帮助用户理解模型，这种解法是目前 LinearRegression 的默认解法。不过有个限制就是样本 feature 的维度不能超过 4096，但对样本数目没有限制。后者就是传统机器学习的解法，是通过迭代寻找最优解的方法。

而且目前 MLlib 也在进一步研究 IterativelyReweightedLeastSquares(IRLS)算法，然后结合 WLS 和 IRLS 就可以使 Spark 支持类似 R GLM 的大多数功能。这样对于前面所有的三种模型都将提供两种解法，用户可以根据实际情况选择合适的解法。这个全部做完预计要在下一个版本 Spark 2.0 了。

2.ALS 算法可以说是目前 MLlib 被应用最多的算法，ALS 的数学原理其实比较容易理解，但是如何在分布式系统中高效实现是一个比较复杂的问题。MLlib 里使用分块计算的思路，合理的设计数据分区和 RDD 缓存来减少数据交换，有效的降低了通信复杂度。这个算法的实现思路充分说明了一个问题：同样的算法，在分布式系统上实现时，不同的选择会带来性能上巨大的差异。

3. 目前的主要的分类模型都支持使用 predictRaw, predictProbability, predict 分别可以得到原始预测、概率预测和最后的分类预测。同时这些分类模型也支持通过设置 thresholds 指定各个类的阈值。不过目前这些预测函数还只支持批量预测，也就是对一个 DataFrame 进行预测，不支持对单个 instance 进行预测。不过单个 instance 的预测的支持也已经在 roadmap 中了。

4.RandomForestClassificationModel 和 RandomForestRegressionModel 模型都支持输出 feature importance

5.GMM EM 算法实现了当 feature 维度或者 cluster 数目比较大的时候的分布式矩阵求逆计算。实验表明当 feature 维度 >30，cluster 数目 >10 的时候，这个优化性能提升明显。

6. 在深度学习方面，MLlib 已经实现了神经网络算法 MultilayerPerceptronClassifier(MLPC)。 这是一个基于前馈神经网络的分类器，它是一种在输入层与输出层之间含有一层或多层隐含结点的具有正向传播机制的神经网络模型，中间的节点使用 sigmoid (logistic) 函数，输出层的节点使用 softmax 函数。输出层的节点的数目表示分类器有几类。MLPC 学习过程中使用 BP 算法，优化问题抽象成 logistic loss function 并使用 L-BFGS 进行优化。

7. 除了我们经常说的机器学习和数据挖掘算法，MLlib 在统计检验算法上也有很大的进步。例如 A/B test, Kolmogorov－Smirnov 检验等。A/B 测试可以说

是很多大数据应用的基础，很多结论最终都是通过 A/B 测试得到的。

机器学习 Pipeline

MLlib 最大的变化就是从一个机器学习的 library 开始转向构建一个机器学习工作流的系统，这些变化发生在 ML 包里面。MLlib 模块下现在有两个包：MLlib 和 ML。ML 把整个机器学习的过程抽象成 Pipeline，一个 Pipeline 是由多个 Stage 组成，每个 Stage 是 Transformer 或者 Estimator。

以前机器学习工程师要花费大量时间在 training model 之前的 feature 的抽取、转换等准备工作。ML 提供了多个 Transformer，极大提高了这些工作的效率。在 1.6 版本之后，已经有了 30+ 个 feature transformer，像 OneHotEncoder, StringIndexer, PCA, VectorSlicer, VectorAssembler, SQLTransformer, HashingTF, IDF, Word2Vec 等。这些工具使得各种 feature 提取、转换等准备工作。

原始数据经过上面的各种 feature transformer 之后就可以丢到算法里去训练模型了，这些算法叫 Estimator，得到的模型是 Transformer。上一章节已经提到了主流的算法支持，所以现在可以得到模型了。得到的模型怎么评价好坏呢？MLlib 提供了像 BinaryClassificationEvaluator 等一系列的 evaluation 工具，这些工具里面定义了像 AUC 等一系列的指标用于评价模型的好坏。

这样一个机器学习的 pipeline 就可以跑起来了，MLlib 进一步提供了像 CrossValidator 这样的工具帮助我们做模型和 pipeline 的调优，选出最佳的模型参数等。另外一个重要特性就是 pipeline 的持久化。现在的 ML pipeline 和大多数的 Transformers/Estimators 都支持了持久化，可以 save/load。这样就可以把模型或者 pipeline 存储、导出到其他应用，扫除了 MLlib 在生产环境应用的最后一个障碍。

另外一个显著变化就是 ML 框架下所有的数据源都是基于 DataFrame，所有的

模型也尽量都基于 Spark 的数据类型 (Vector, Matrix) 表示。在 ML 里面的 public API 下基本上看不到对 RDD 的直接操作了，这也与前面讲的 Spark 的未来变化是一致的。

Python 是机器学习领域最流行的语言，ML/MLlib 的 Python API 也在不断加强，越来越多的算法和功能的 Python API 基本上与 Scala API 对等了。

SparkR

R 语言在统计领域应用非常广泛，R 语言本身的架构可以比较容易支持其他执行后端。SparkR 就是把 Spark 作为 R 语言的执行后端。目前 SparkR 提供的接口包括了 DataFrame 的绝大多数函数以及机器学习 GLM 函数，未来还会支持更多的功能。SparkR 既可以利用 R 接口的易用性以及在传统行业的既有市场空间，又可以利用 Spark 强大的分布式数据处理能力。SparkR 在推出不久就获得了很高的用户关注度和使用率。笔者在和一些传统行业的大数据从业者交流中经常会被问到这样的问题：我以前用 R 写的程序，现在数据量大了，传统 R 跑不动了，能不能直接放到 Spark 上跑。相信 SparkR 就是在朝着解决这个问题的方向努力。

在统计和机器学习方面一个重要的 feature 就是 R formula 的支持，R 用户可以用他们非常熟悉的 formula 来定义一个 GLM 模型。目前已经支持最基本的 '.', '~', ':', '+' 和 '-'，未来还会增强这项功能。同时 SparkR 的 GLM 函数不只提供了训练一个 GLM 模型和预测的能力，也提供了对模型的 R 类似的统计指标输出。目前 SparkR 跑出的 GLM 模型和传统 R 跑出的模型的结果是一样的，同时也会输出 coefficients standard errors, p-values, t-values 等统计信息。

Spark 2.0

Spark streaming 等组件在这一年也有很大的变化，笔者由于精力有限，在此不一一列出。Spark 2.0 预计三四月份发布，将会确立以 DataFrame 和 Dataset 为

核心的体系架构，RDD 会慢慢退出历史舞台；同时在各方面的性能上会有很大的提升，当然我们也期待着稳定性方面的提升。从 1.x 到 2.x 还会放弃 Hadoop 1.x 的支持，RPC 系统从 Akka 迁移到 Netty 等。快速发展的社区，越来越多的应用，性能和稳定性方面的不断提升使得 Spark 在未来的若干年内还是大数据处理工具的首选。

作者简介：梁堰波，明略数据技术合伙人，开源爱好者，Apache Spark 项目核心贡献者。北京航空航天大学计算机硕士，曾就职于 Yahoo!、美团网、法国电信从事机器学习和推荐系统相关的工作，在大数据、机器学习和分布式系统领域具备丰富的项目经验。

程序员的
地球漫游指南

《神秘的程序员们》之

那些令你憎恶的系统是怎么来的？

by 西乔

扫码关注
微信公众号
coderstory
原创首发

什么垃圾系统，都这年头了
每天晚上还停机维护6个小时！

普通用户

这样过了好几年，
用户们终于等到了这一天

最后一个季度了，
得把预算用一个稳妥的方式赶紧花掉。

甲方
CIO

咳咳，我们的系统现在需要频繁停机维护。我看很多用户抱怨，这问题应该好好解决下，加快建设，顺势而为，提升用户体验，要与时代同步呐。

您说得太好了，我们一定完全按照贵部门的需求来规划和完成项目。

（乙方）
咨询公司
项目经理

只要能让甲方满意、通过验收，你们想整成啥样就啥样。

我们系统里有十几年的数据，
有很多服务都在调用这些数据。
这次先不要对数据库做升级。

甲方
DBA

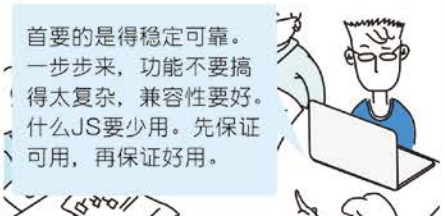
这么多年了，天知道历史数据里有啥毛病，反正我在任的时候不能让他们动数据。

动作越大
越危险

预算和时间都有限，
一定要稳妥。整体的
架构最好不要动。

甲方
技术主管

首要的是得稳定可靠。一步步来，功能不要搞得太复杂，兼容性要好。什么JS要少用。先保证可用，再保证好用。



嗯，保守一点的话风险小责任少，交接过来后也容易维护。

甲方总工

《神秘的程序员们》是国内首部以程序员、技术文化、项目管理及互联网创业为主题的原创漫画。始于09年，曾于《程序员》杂志上连载5年时间，拥有上百万读者。现于微信公众平台上重新开始发布连载。

程序员的
地球漫游指南

《神秘的程序员们》 之

那些令你憎恶的系统是怎么来的？

by 西乔

扫码关注
微信公众号
coderstory
原创首发

InfoQ 的读者们：

很高兴能在InfoQ的平台上继续刊登《神秘的程序员们》漫画。InfoQ是我多年以来一直非常欣赏的技术媒体平台，长期坚持创办高品质的活动和内容，在技术嗅觉、视野和务实性上一直走在前列。几年前我就给第一届ArchSummit的深圳会画过漫画海报，而15年底我有幸在最近的这届北京会场待了两天，有幸得到InfoQ的提供一个窗口能让我和读者有深入的互动及交流。希望未来能有更多这样的合作，让各类技术活动、技术内容能更有趣、创意和多元化。

最后祝愿所有的技术人在2016年都能实现自己的价值、忠于自我、感受幸福，依然坚持走在追求梦想追求技术之美的道路上。

解读大数据：大数据的黄金时代

作者 杜小芳

本文是大数据解读篇，在这篇文章里我们将回顾 2015 展望 2016，看看过去的一年里广受关注的技术有哪些进展，了解下数据科学家这个职业的火热。 在关键技术进展部分我们在大数据生态圈众多技术中选取了 Hadoop、Spark、Elasticsearch 和 Apache Kylin 四个点，分别请了四位专家：Hulu 的董西成、明略数据的梁堰波、精硕科技的卢亿雷、eBay 的韩卿，来为大家解读 2015 里的进展。

回顾 2015 年的关键技术进展

Hadoop

Hadoop 作为大数据平台中最基础与重要的系统，在 2015 年提高稳定性的同时，发布了多个重要功能与特性，这使得 Hadoop 朝着多类型存储介质和异构集群的方向迈进了一大步。

HDFS

HDFS 之前是一个以磁盘单存储介质为主的分布式文件系统。但随着近几年新存储介质的兴起，支持多存储介质早就提上了日程。如今，HDFS 已经对多存储介质有了良好的支持，包括 Disk、Memory 和 SSD 等，对异构存储介质的支持，使得 HDFS 朝着异构混合存储方向发展。目前 HDFS 支持的存储介质如下：

- ARCHIVE：高存储密度但耗电较少的存储介质，通常用来存储冷数据。
- DISK：磁盘介质，这是 HDFS 最早支持的存储介质。
- SSD：固态硬盘，是一种新型存储介质，目前被不少互联网公司使用。
- RAM_DISK ：数据被写入内存中，同时会往该存储介质中再（异步）写一份。

YARN

YARN 作为一个分布式数据操作系统，主要作用是资源管理和资源调度。在过去一年，YARN 新增了包括基于标签的调度、对长服务的支持、对 Docker 的支持等多项重大功能。

基于标签的调度，使得 YARN 能够更好地支持异构集群调度。它的基本思想是，通过打标签的方式为不同的节点赋予不同的属性，这样，一个大的 Hadoop 集群按照节点类型被分成了若干个逻辑上相互独立（可能交叉）的集群。这种集群跟物理上独立的集群很不一样，用户可以很容易地通过动态调整 label，实现不同类型节点数目的增减，这具有很好的灵活性。

对长服务的支持，使得 YARN 逐渐变为一个通用资源管理和调度系统。目前，YARN 既支持像类似 MapReduce，Spark 的短作业，也支持类似 Web Service，MySQL 这样的长服务。 支持长服务是非常难的一件事情，YARN 需要解决以下问题：服务注册、日志滚动、ResourceManager HA、NodeManager HA（NM 重启过程中，不影响 Container）和 ApplicationMaster 永不停止，重启后接管之前的 Container。截止 2.7.0 版本，以上问题都已经得到了比较完整的解决。

对 Docker 的支持，使得 YARN 能够为上层应用提供更好的打包、隔离和运行方式。YARN 通过引入一种新的 ContainerExecutor，即 DockerContainerExecutor，实现了对 Docker 的支持，但目前仍然是 alpha 版本，不建议在生产环境中使用。

HBase

在 2015 年，HBase 迎来了一个里程碑——HBase 1.0 release，这也代表着 HBase 走向了稳定。 HBase 新增特性包括：更加清晰的接口定义，多 Region 副本以支持高可用读，Family 粒度的 Flush 以及 RPC 读写队列分离等。

Spark

2015 年的 Spark 发展很快，JIRA 数目和 PR 数目都突破了 10000，contributors 数目超过了 1000，可以说是目前最火的开源大数据项目。这一年 Spark 发布了多个版本，每个版本都有一些亮点：

- 2014 年 12 月，Spark 1.2 发布引入 ML pipeline 作为机器学习的接口。
- 2015 年 3 月，Spark 1.3 发布引入了 DataFrame 作为 Spark 的一个核心组件。
- 2015 年 6 月，Spark 1.4 发布引入 R 语言作为 Spark 的接口。R 语言接口在问世一个多月之后的调查中就有 18% 的用户使用。
- 2015 年 9 月，Spark 1.5 发布。Tungsten 项目第一阶段的产出合并入 DataFrame 的执行后端，DataFrame 的执行效率得到大幅提升。
- 2016 年 1 月，Spark 1.6 发布引入 Dataset 接口。

Spark 目前支持四种语言的接口，除了上面提到的 R 语言的使用率以外，Python 的使用率也有很大提升，从 2014 年的 38% 提升到 2015 年的 58%；而 Scala 接口的使用率有所下降，从 84% 下降到 71%。同时 Spark 的部署环境也有所变化，51% 的部署在公有云上，48% 使用 standalone 方式部署，而在 YARN 上的只有 40% 了。可见 Spark 已经超越 Hadoop，形成了自己的生态系统。而在形成 Spark 生态系统中起到关键作用的一个 feature 就是外部数据源支持，Spark 可以接入各种数据源的数据，然后把数据导入 Spark 中进行计算、分析、挖掘和机器学习，

然后可以把结果在写出到各种各样的数据源。到目前为止 Spark 已经支持非常多的外部数据源，像 Parquet/JSON/CSV/JDBC/ORC/HBase/Cassandra/Mongodb 等等。

上面这些调查数据来自美国，中国的情况有所区别，但是还是有一定的借鉴意义的。国内的 Spark 应用也越来越多：腾讯的 Spark 规模到了 8000+ 节点，日处理数据 1PB+。阿里巴巴运行着目前最长时间的 Spark Job：1PB+ 数据规模的 Spark Job 长达 1 周的时间。百度的硅谷研究院也在探索 Spark+Tachyon 的应用场景。

Spark MLlib 的 ALS 算法已经在很多互联网公司用于其推荐系统中。基本上主流的互联网公司都已经部署了 Spark 平台并运行了自己的业务。上面说的更多的互联网的应用，实际上 Spark 的应用场景有很多。在 Databricks 公司的调查中显示主要应用依次是：商务智能、数据仓库、推荐系统、日志处理、欺诈检测等。

除了互联网公司以外，传统 IT 企业也把 Spark 作为其产品的一个重要组成。IBM 在今年 6 月的 Spark summit 期间宣布重点支持 Spark 这个开源项目，同时还开源了自己的机器学习系统 SystemML 并推进其与 Spark 的更好合作。美国大数据巨头 Cloudera，Hortonworks 和 MapR 都表示 Spark 是其大数据整体解决方案的核心产品。可以预见 Spark 是未来若干年最火的大数据项目。

在深度学习方面 2015 年可谓非常热闹，如 Google 开源其第二代机器学习系统 TensorFlow，Facebook 开源 Torch 和人工智能硬件服务器 Big Sur 等等。Spark 社区也不甘落后，在 1.5 版本中发布了一个神经网络分类器 MultiplayerPerceptronClassifier 作为其深度学习的雏形。虽然这个模型还有很多地方需要优化，大家不妨尝试下，毕竟它是唯一一个基于通用计算引擎的分布式深度学习系统。

除了现在非常火的深度学习，在传统统计和机器学习领域，Spark 这一年也有非常大的变化，包括 GLM 的全面支持，SparkR GLM 的支持，A/B test，以及像 WeightesLeastSquares 这样的底层优化算法等。

具体内容可以看梁堰波在 InfoQ 上的年终回顾：《解读 2015 之 Spark 篇：新生态系统的形成》。

Elasticsearch

Elasticsearch 是一个可伸缩的开源全文搜索和分析引擎。它可以快速地存储、搜索和分析海量数据。Elasticsearch 基于成熟的 Apache Lucene 构建，在设计时就是为大数据而生，能够轻松的进行大规模的横向扩展，以支撑 PB 级的结构化和非结构化海量数据的处理。Elasticsearch 生态圈发展状态良好，整合了众多外围辅助系统，如监控 Marvel，分析 Logstash，安全 Shield 等。近年来不断发展受到广泛应用，如 Github、StackOverflow、维基百科等，是数据库技术中倍受关注的一匹黑马。

Elasticsearch 在今年下半年发布了 2.0 版本，性能提升不少，主要改变为：

- Pipeline Aggregation：流式聚合，像管道一样，对聚合的结果进行再次聚合。原来 client 端需要做的计算工作，下推到 ES，简化 client 代码，更容易构建强大的查询。
- Query/Filter 合并：取消 filters，所有的 filter 语句自动转换为 query 语句。在上下文语义是 query 时，进行相关性计算；上下文语义是 filter 时，简单排除 b 不匹配的 doc，像现在的 filter 所做的一样。这个重构以为着所有的 query 执行会以最 有效的顺序自动优化。例如，子查询和地理查询会首先执行一个快速的模糊步骤，然后用一个稍慢的精确 步骤截断结果。在 filter 上下文中，cache 有意义时，经常使用的语句会被自动缓存。
- 可配置的 store compression：存储的 field，例如 _source 字段，可以使用默认的 LZ4 算法快速压缩，或者使用 DEFLATE 算法减少 index size。对于日志类的应用尤其有用，旧的索引库在优化前可以切换到 best_compression。
- Hardening：Elasticsearch 运行于 Java Security Manager 之下，在安全性上标志着一个巨大的飞跃。Elasticsearch 难于探测，黑客在系统上 的影响也被严格限制。在索引方面也有加强： indexing 请求 ack

前，doc 会被 fsync，默认写持久化 所有的文件都计算 checksum，提前检测文件损坏 所有的文件 rename 操作都是原子的（atomic），避免部分写文件 对于系统管理员来讲，一个需求较多的变化是，可以避免一个未配置的 node 意外加入 Elasticsearch 集群网络：默认绑 定 localhost only， multicast 也被移除，鼓励使用 unicast。

- Performance and Resilience：除上所述，Elasticsearch 和 Lucene 还有很多小的变化，使其更加稳定可靠，易于配置，例如：默认 doc value，带来更少的 heap usage，filter caching 更多使用 bitsets type mappings 大清理，更安全可靠，无二义性 cluster stat 使用 diff 进行快速变化传播，带来更稳定的大规模集群
- Core plugins：官方支持的 core plugins 同时发布，和 Elasticsearch 核心使用相同的版本号。
- Marvel 2.0.0 free to use in production：Marvel 免费。

Apache Kylin

Apache Kylin 是一个开源的分布式分析引擎，提供 Hadoop 之上的 SQL 查询接口及多维分析（OLAP）能力以支持超大规模数据，最初由 eBay Inc. 开发并贡献至开源社区。最初于 2014 年 10 月 1 日开源，并于同年 11 月加入 Aapche 孵化器项目，并在一年后的 2015 年 11 月顺利毕业成为 Apache 顶级项目，是 eBay 全球贡献至 Apache 软件基金会（ASF）的第一个项目，也是全部由在中国的华人团队整体贡献至 Apache 的第一个项目。

在 eBay，已经上线两个生产环境平台，有着诸多的应用，包括用户行为分析、点击分析、商户分析、交易分析等应用，最新的 Streaming 分析项目也已经上线。目前在 eBay 平台上最大的单个 cube 包含了超过 1000 亿的数据，90% 查询响应时间小于 1.5 秒，95% 的查询响应时间小于 5 秒。同时 Apache Kylin 在 eBay 外部也有很多的用户，包括京东、美团、百度地图、网易、唯品会、Expedia、Expotional 等很多国内外公司也已经在实际环境中使用起来，把 Apache Kylin 作为他们大数据分析的基础之一。

过去的一年多是 Apache Kylin 发展的重要的一年：

- 2014 年 10 月 1 日，Kylin 代码在 github.com 上正式开源
- 2014 年 11 月 25 日，正式加入 Apache 孵化器并正式启用 Apache Kylin 作为项目名称
- 2015 年 6 月 10 日，Apache Kylin v0.7.1-incubating 发布，这是加入 Apache 后的第一个版本，依据 Apache 的规范作了很多修改，特别是依赖包，license 等方面，同时简化了安装，设置等，并同时提供二进制安装包
- 2015 年 9 月 6 日，Apache Kylin v1.0-incubating 正式发布，增强了 SQL 处理，提升了 HBase coprocessor 的性能，同时提供了 Zeppelin Interpreter 等
- 2015 年 9 月 16 日，Apache Kylin 与 Spark，Kafka，Storm，H2O，Flink，Elasticsearch，Mesos 等一起荣获 InfoWorld Bossie Awards 2015：最佳开源大数据工具奖，这是业界对 Kylin 的认可
- 2015 年 11 月 18 日，Apache Kylin 正式毕业成为 Apache 顶级项目
- 2015 年 12 月 15 日，Apache Kylin v1.2 正式发布，这是升级为顶级项目后的第一个版本，提供了对 Excel，PowerBI，Tableau 9 等的支持，对高基维度增强了支持，修复了多个关键 Bug 等
- 2016 年，Apache Kylin 将迎来重要的 2.x 版本，该版本对底层架构和设计作了重大重构，提供可插拔的设计及 Lambda 架构，同时提供对历史数据查询，Streaming 及 Realtime 查询等，同时在性能，任务管理，UI 等各个方面提供增强。

同时，过去一年也是社区发展的重要一年，在过去一年内发展了来自 eBay，美团，京东，明略数据，网易等众多 committer，社区每天的讨论也是非常热闹。社区提交了很多新特性和 Bug 修复，包括来自美团的不同 HBase 写入，来自京东的明细数据查询，来自网易的多 Hive 源等多个重大特性为 Apache Kylin 带来了巨大的增强。

社区合作

在开源后的一年时间内，Apache Kylin 也和其他社区建立了良好的合作关系，

Apache Calcite 作为 Kylin 的 SQL 引擎被深入的整合进来，我们也向 Calcite 提交了很多改进和修复，Calcite 的作者，Julian Hyde 也是 Kylin 的 mentor。HBase 是 Kylin 的存储层，在实际运维中，我们碰到过无数问题，从可靠性到性能到其他各个方面，Kylin 社区和 HBase 社区积极合作解决了绝大部分关键问题。另外，现在越来越多的用户考虑使用 Apache Zeppelin 作为前端查询和展现的工具，为此我们开发了 Kylin Interperter 并贡献给了 Zeppelin，目前可以直接从最新版的 Zeppelin 代码库中看到这块。同样，我们也和其他各个社区积极合作，包括 Spark，Kafka 等，为构建和谐的社区氛围和形成良好合作打下了坚实的基础。

技术发展

技术上，这一年来 Apache Kylin 主要在以下几个方面。

Fast Cubing：在现在的版本中，Cube 的计算依赖 MapReduce，并且需要多个步骤的 MR Job 来完成计算，且 MR Job 的多少和维度相关，越多的维度会带来更多的 MR job。而每一次 MR job 的启停都需要等待集群调度，并且 MR job 之间的数据需要多次在 HDFS 落地和传输，从而导致消耗了大量的集群资源。为此我们引入了一种新的算法：Fast Cubing。一个 MapReduce 即可完成 Cub 的计算，测试结果表明整个 Cubing 的时间可以降低 30 ～ 50% 左右，网络传输可以下降 5 倍，这在超大规模数据集的计算上带来了客观的性能改进。

Streaming OLAP：Kylin 作为一个预计算系统，不可避免的有着数据刷新延迟的限制，这在大部分用户案例中并不是问题，但随着业务和技术的发展，Streaming 甚至 Realtime 的需求越来越高。2015 年 Kylin 的主要发展都在 Streaming OLAP 上，为了支持低延迟的数据刷新，从整体的架构和设计上都做了相当大的重新设计，目前已经可以支持从 Kafka 读取数据并进行聚合计算的能力，同时提供 SQL 接口为前端客户端提供标准的访问接口，数据延迟已经可以做到分钟级别。

Spark Cubing：Spark 作为 MapReduce 的一种替代方案一直在社区中被问及 Kylin 是否可以支持直接使用 Spark 来作为计算。为此我们在 2015 年下半年实

现了同样算法的 Spark Cubing 引擎，目前还在测试中。

可插拔架构：为了更广泛的可扩展性，并支持如上各种新特性，Kylin 在 2.x 的代码中引入了可插拔架构和设计，从而解决了对特定技术的依赖问题。在新的设计中，数据源可以从 Hive，SparkSQL 等各种 SQL on Hadoop 技术读取，并支持 Kafka；在计算引擎方面，除了 MapReduce 方面的 Fast Cubing 外，实现了 Spark Cubing，Streaming Cubing 等多种计算框架，并为将来其他计算框架留下了扩展接口；在存储上，HBase 目前依然是唯一的存储层，但在上层设计中已经很好的进行了抽象，很容易可以扩展到其他 Key — Value 系统。

大数据与机器学习

机器学习是数据分析不可缺少的一部分。机器学习被赞誉为大数据分析和商务智能发展的未来，成功的机器学习项目依赖于很多因素，包括选择正确的主题，运行环境，合理的机器学习模型，最重要的是现有的数据，大数据为机器学习提供了很好的用武之地。

机器学习正很快从一个被很少人关注的技术主题转变为被很多人使用的管理工具。优秀的算法，大数据和高性能的计算资源的条件的满足使得机器学习快速发展，机器学习在今年第一次进入 Gartner 技术成熟曲线的报告中，并且进入大数据一样的应用期；而机器学习也是报告中第一个出现的技术。2015 年是机器学习丰收年，发生了很多令人瞩目的大事。

各大巨头开源：

- 2015 年 1 月，Facebook 开源前沿深度学习工具“Torch”。
- 2015 年 4 月，亚马逊启动其机器学习平台 Amazon Machine Learning，这是一项全面的托管服务，让开发者能够轻松使用历史数据开发并部署预测模型。
- 2015 年 11 月，谷歌开源其机器学习平台 TensorFlow。
- 同一月，IBM 开源 SystemML 并成为 Apache 官方孵化项目。
- 同时，微软亚洲研究院将分布式机器学习工具 DMTK 通过 Github 开源。

DMTK 由一个服务于分布式机器学习的框架和一组分布式机器学习算法组成，可将机器学习算法应用到大数据中。

- 2015 年 12 月，Facebook 开源针对神经网络研究的服务器“Big Sur”，配有高性能图形处理单元（GPUs），转为深度学习方向设计的芯片。

大公司不仅是用开源社区来增强自己的机器学习工具，而且也会以收购来提升自身的机器学习实力。如 IBM 于今年 3 月收购了 AIchemyAPI，AIchemyAPI 能够利用深度学习人工智能，搜集企业、网站发行的图片和文字等来进行文本识别和数据分析。

此外，2015 年不仅仅是关于大公司的，利用机器学习的各种创业公司也占了同等地位。比如 EverString 完成 B 轮融资，该公司利用企业内部销售数据，和不断主动挖掘分析全球新闻数据，社交媒体等外部数据，通过机器学习自动建立量化客户模型，为企业预测潜在客户。

数据科学家的崛起

大数据需要数据分析，数据分析需要人才。数据科学是早就存在的词汇，而数据科学家却是近年来突然出现的新词。在 Google、Amazon、Quora、Facebook 等大公司的背后，都有一批数据科学专业人才，将大量数据变为可开发有价值的金矿。在大数据时代，数据科学家等分析人才的需求在激增。

据相关报告，国内大数据人才缺口目前已达百万，一名高级数据挖掘工程师月薪高达 30K-50K。招聘网站上的每天都会产生大量的大数据相关职位需求。据拉勾网提供的统计来看，从 2014 年到 2015 年，IT 行业关于大数据的岗位需求增长了 2.4 倍。人才培养迫在眉睫。复旦大学于今年成立了全国首个大数据学院。阿里云于年底宣布新增 30 所合作高校，开设云计算大数据专业，计划用 3 年时间培养 5 万名数据科学家。各知名大学也将数据科学设为硕士课程。

无论是国内还是国外，数据科学都是目前炙手可热的研究领域，数据科学家、

数据分析师都是非常火爆的职位，几乎所有的产业都需要数据科学家来从大量的数据中挖掘有价值的信息。大数据分析领域的专属首席级别头衔也愈发多见。美国政府今年任命了 DJ Patil 作为政府的首席数据科学家（Chief Data Scientist），这也是美国政府内部首次设立“数据科学家”这个职位。

展望 2016

- Hadoop。对于 HDFS，会朝着异构存储介质方向发展，尤其是对新兴存储介质的支持；对于 YARN，会朝着通用资源管理和调度方向发展，而不仅仅限于大数据处理领域，在加强对 MapReduce、Spark 等短类型应用支持的同时，加强对类似 Web Service 等长服务的支持；
- 对于 HBase，将会花费更多精力在稳定性和性能方面，正尝试的技术方向包括：对于 HDFS 多存储介质的使用；减少对 ZooKeeper 的使用以及通过使用堆外内存缓解 Java GC 的影响。
- Spark 2.0 预计明年三四月份发布，将会确立以 DataFrame 和 Dataset 为核心的体系架构。同时在各方面的性能上会有很大的提升。
- Apache Kylin 2.0 即将发布，随着各项改进的不断完善，该版本将在 2016 年在 OLAP on Hadoop 上更进一步！
- Elasticsearch 开源搜索平台，机器学习，Data graphics，数据可视化在 2016 年会更加火热。
- 大数据会越来越大，IOT、社交媒体依然是一个主要的推动因素。
- 大数据的安全和隐私会持续受到关注。

专家介绍

董西成，就职于 Hulu，专注于分布式计算和资源管理系统等相关技术。《Hadoop 技术内幕：深入解析 MapReduce 架构设计与实现原理》和《Hadoop 技术内幕：深入解 析 YARN 架构设计与实现原理》作者，dongxicheng.org 博主。

梁堰波，明略数据技术合伙人，开源爱好者，Apache Spark 项目核心贡献者。北京航空航天大学计算机硕士，曾就职于 Yahoo!、美团网、法国电信从事机器

学习和推荐系统相关的工作，在大数据、机器学习和分布式系统领域具备丰富的项目经验。

卢亿雷，精硕科技（AdMaster）技术副总裁兼总架构师，大数据资深专家，CCF（中国计算学会）大数据专委委员，北航特聘教授。主要负责数据的采集、清洗、存储、挖掘等整个数据流过程，确保提供高可靠、高可用、高扩展、高性能系统服务，提供 Hadoop/HBase/Storm/Spark/ElasticSearch 等离线、流式及实时分布式计算服务。对分布式存储和分布式计算、超大集群、大数据分析等有深刻理解及实践经验。有超过 10 年云计算、云存储、大数据经验。曾在联想、百度、Carbonite 工作，并拥有多篇大数据相关的专利和论文。

韩卿（Luke Han），eBay 全球分析基础架构部（ADI） 大数据平台产品负责人，Apache Kylin 副总裁，联合创始人，管理和驱动着 Apache Kylin 的愿景，路线图，特性及计划等，在全球各地不同部门中发展客户，开拓内外部合作伙伴及管理开源社区等，建立与大数据厂商，集成商及最终用户的联系已构建健壮的 Apache Kylin 生态系统。在大数据，数据仓库，商务智能等方面拥有超过十年的工作经验。

解读容器：扩张与进化

作者 张磊

回顾 2015 年容器技术的发展历程，我们可以用两个关键词来概括：扩张与进化。

如果说 2014 年仅仅是 Docker 为主的容器技术在云计算以及 DevOps 圈初露锋芒的话，那么 2015 年则是以 Docker 为核心的容器生态圈迅猛扩张的一年。这种扩张的态势，一直从 2015 上半年火爆的 DockerCon 蔓延到了 2015 年的最后一天。伴随着容器技术快速发展的过程，国内外的技术人员有幸亲历了 OCI、CNCF 两大标准组织的确立，第一时间体验了 Docker 1.8 和 1.9 两大关键版本的发布，见识到了 CoreOS 一系列关键技术革新与战略布局，也感受到了国内 Docker 创业圈的如火如荼。

国外容器技术项目

在 2015 年，一直以“善意独裁”面孔示人的 Docker 公司终于迈出了合作的第一步。OCI 组织的成立宣告了工业界对 Docker 提出的容器技术的高度认同，也

暗含了后进场玩家试图从这个由 startup 开创的新领域中分得一杯羹的决心。然而 runC 项目运作到现在，依然没能够进入 Docker Daemon 的实现主干，也没有任何巨头发布以 runC 为基础的新容器实现。Docker 而不是 runC 被用户当作容器技术的事实标准的现状在短期内（1-2 年）恐怕还很难发生本质改变。容器技术领域多样化的任务目前还是落在直接竞争对手 CoreOS，以及另辟蹊径的虚拟化容器技术比如 Hyper 和 Intel Clear Linux 身上。但是无论如何，诞生于 startup 的 Docker 容器注定要经历更多的考验才能在巨头林立的云计算领域真正地扎根生长，无论其是否愿意，将容器技术标准化是这条道路上必须面对的选择和进化方向。

另一方面，Docker 公司这种独一无二的亲和力和号召力正是 2015 年 Docker 为主的容器生态圈版图迅速扩张的主要基石。当然，既然是扩张，这张容器生态圈版图的背后也必然少不了大小领主“封地”的确立和斗争。

2015 年，Google 和 RedHat 联盟以 Kubernetes 1.0 为阵地宣告了大规模容器编排与管理领域的领主地位。号称以 Borg 等 Google 多年容器技术实践经验为理论指导、以 Google 和 RedHat PaaS 团队为主要工程力量的 Kubernetes 项目一经宣布生产级别可用，便立刻吸引了的工业界乃至学术界的大量关注和投入。尽管不是第一家，但是我们不得不承认 Google 的号召力和 Kubernetes 不凡的背景直接推动了 CNCF 这一容器编排管理标准组织的诞生。在技术方向上，Kubernetes 团队则试图以 Kubernetes 为依托来对外输出 Google 的容器技术的思想和经验，或多或少有点要弥补 LMCTFY 项目中途夭折的意思。无论如何，Kubernetes 所体现出的前瞻性的容器技术实践思路，确实值得我们每一个容器技术实践者重点关注和学习。无论是 Pod 及伙伴容器、单 Pod 单 IP 网络模型、volume 插件机制、容器生命周期钩子这种对容器技术本身的改造，还是虚拟 IP 与负载均衡、垂直健康检查、密码数据卷管理、元数据 API 等平台级别能力的实现，都展现出了 Kubernetes 与其他编排管理项目与众不同的技术视野和团队实力。在未来发展方向上，Kubernetes 已经开始向 1000+ 节点的集群规模进发，毕竟在性能和规模化领域，Kubernetes 没有理由落后竞争者太久。另一方面，除了常规的 long time running 任务，其他类型任务比如短任务和 daemon 任务的支持都已经引入了项目主干，类似 big data 业务的支持将是未来的一个重要方向。

在与 Docker”分手“之后，CoreOS 一直在积极地寻求展示自己技术想法的机会，包括加入 OCI 组织以及在 Kubernetes 上同 Google 开展深度的合作。鉴于 OCI 最开始只关注于 container runtime 的实现标准，CoreOS 的 AppC 一直在积极推进镜像标准的概念。目前这个标准已经在 Kubernetes 上初见雏形。最值得关注的是，CoreOS 还在 0.8.0 版本发布时高调宣布了同 Intel Clear Linux 团队合作在 rkt 上实现了基于虚拟化技术隔离的容器（这与国内的 Hyper 团队的做法不谋而合，只不过后者是在 Docker 上做出的类似实现）。这种 hypervisor-based container 是目前在公有云上提供容器服务最佳选择，CoreOS 在容器安全和隔离性问题上进行本质革新的做法的确很有说服力。而在容器编排管理领域，CoreOS 公司将 Kubernetes 组件内置到 CoreOS 项目中作为主要的底层依赖之一。Etcd 的作者目前也正在 Kubernetes 社区积极推进项目整体性能提升和调度效率优化的工作，毕竟作为整个项目的核心依赖，Etcd 的作者们有足够的理由和能力承担起 Kubernetes 性能提升的重任。这一点上其他容器管理项目可能要稍微眼红一下了。

而在另一边，Mesosphere 公司借助在资源调度管理领域与生俱来的优势，在 2015 年成功开拓出了一片以 DCOS 为核心、兼顾大数据和应用托管的服务平台市场。Apache Mesos 项目原生的两级调度和多框架支持使得用户在自己的组织内部设施云计算平台终于变得唾手可得。尤其是在传统技术型企业转型互联网架构的过程中，Mesos 生态圈能够非常方便的在企业内部迅速实现一套原本在一线互联网公司中都算得上“黑科技”的资源调度管理平台，然后快速搭建起 PaaS 和 BigData 服务。尽管 Mesos 原生并非针对容器设计，Mesosphere 所提供的诸如 Marathon 等上层解决方案也明显在成熟度和技术实现上有着这样那样的不足，但是不得不说 Mesos 生态体系目前是企业自建私有云最快速、最有利于展示 POC 的一套技术方案。鉴于 Mesos 本身较难只针对 Docker 进行根本性的改造，Mesosphere 生态圈目前依赖于 Marathon 等上层项目来响应 Docker 容器技术的快速发展，在这种形势下，一组包含了用户生命周期管理、多维度监控、整合大数据业务管理等功能的完善的 PaaS 很可能是这些上层框架的最终形态。

当然，最后一定要说的就是 Docker 公司了。在进入 2015 年之后，雄心勃勃的 Docker 公司在 Docker 源码层面开始了大规模的重构，将原先仓促发布的很多模

块进行了统一的抽象和整合，使得在 1.9 版本彻底解耦 Volume 和 Network 成为了可能。另一方面，Docker 公司加紧了自己在组建生态圈闭环上的战略布局，这一点以年末收购 Tutum 为 2015 年画上了圆满的句号。之所以强调这一点，是因为 Docker 之前的收购对象都是在某一领域具有独创性的公司或团队，比如容器编排上的 Fig，容器网络上的 SocketPlane，而 Tutum 虽然在 Control Panel 以及产品化上做的很早很出色，但是类似的竞争者却不在少数且产品能力也很强，更何况 Docker 公司自己在这一领域已经有所动作。所以 Tutum 最终胜出的确是自身技术和产品实力的最有力证明。

在技术路线上，Docker 把同 runC 的集成列到了高优先级任务上，并且已经为之做了大量重构工作，但是奈何 daemon 同 libcontainer 的交互过于繁杂，此项工作进展一直缓慢。好消息是 Docker 在年末发布了 Containerd 项目来专门负责同 runC 进行交互，此项目最终会进入 Docker Engine 主干从而间接实现 Docker 同 libcontainer 的解耦。一旦这个目的达成，Docker daemon 的复杂度会大大降低，性能也很可能得到大幅提升，更重要的是届时容器开发者将有能力定制自己的 daemon，在容器端加入定制化的功能和特性，这才是 runC 项目的真正意义和玩法，非常值得期待。Docker 公司在 2015 年的另一个大动作便是 1.9 版本的发布终于完成了存储和网络功能的解耦，使得用户可以以插件的方式提供第三方存储和网络功能来支持远程数据卷和跨主机网络。

但是需要提醒读者的是，无论是网络还是存储，这些插件方式的工作原理与非插件方式并没有本质区别，Docker 并没有能力也没有理由提供任何优化。所以在这一点上，普通用户在前期阶段需要寄希望于社区里的插件开发者的能力和技术水平不要太低。因为笔者前面的经验表明即使是 Docker 官方比如 SocketPlane 提供的网络方案，在稳定性、可用性上也没有特别值得表扬的地方，建议用户保守上线该功能，必要时自己按需开发插件。“Docker 之心，路人皆知”。这家已经在云计算领域掀起革命的 startup 背后的野心之大，的确配得上目前它在轻量级应用容器界的号召力和绝对地位。在未来的发展方向上，有如下几个方向上的进化是一定会发生的：

1. Docker Engine 的进一步模块化和解耦，最终用户一定可以选择使用其中的一部分来达成自己的目的；

2. runC 全面取代 execdriver 来执行容器；
3. 更丰富的插件能力和以此为基础的数据卷管理能力（类似 Flocker）。

在容器编排与管理领域，Docker 已经构建出了 Compose+Swarm+Machine 的技术闭环，这套技术栈的最大亮点是全面兼容 Docker API。当然，这个优势的前提是目前 Docker 而不是 runC 仍然是业界公认的容器标准。在这个领域，Docker 目前选择了重点照顾用户友好度而适当放弃性能和规模能力的路线，毕竟在兼容 Docker API 的前提下引入更复杂的编排、调度和管理能力是比较困难的。这也是为什么 Swarm 现在正在推进同 Mesos 集成以提高调度方面的性能和可扩展能力，虽然笔者认为这个路线可能并不太友好（想象一下宿主机上同时运行着 Docker dameon，Swarm agent 和 Mesos Slave 的场景）。

总之，Docker 目前在容器界的领导地位已经毋庸置疑。一系列产品和技术布局的完成也确立了 Docker 公司在这一由它自己开拓的疆土上的霸主地位。接下来，如何在不甘心的巨头们设置的规则丛林里生存、壮大并且健康发展下去，甚至达成 Linux 项目的创举，则是考验这家已经创造了一个不小的奇迹的初创团队真正实力和水平的难题。

国内容器技术圈

与国外相比，以 Docker 为主的容器技术在国内的发展势头甚至要更猛烈一些，其中部分原因是因为 2014 年以前的 PaaS 风潮没能够在国内掀起本质上的变革，使得国内云计算圈子在除了 IaaS 之外的领域颇有点一筹莫展的感觉。在这层意义上，Docker 容器技术的诞生和普及绝对起到了久旱逢甘霖的效果。容器创业组织雨后春笋般萌发，不少团队的背景也跟经典 PaaS 领域息息相关；另一方面原先经典 PaaS 从业者的转型到容器创业领域的也不在少数。所以目前国内 Docker 创业项目的产品形态，有一部分延续了原先 PaaS 项目的产品路线（尤其是 Cloud Foundry），比如：

1. 重点关注应用生命周期管理；
2. 强调应用访问和域名绑定；
3. 纳入 Docker 的部分概念比如数据卷和镜像；

4. 对用户屏蔽网络、存储和调度细节；
5. 将数据库等应用依赖抽象成”服务”。

而另外一部分则选择了稍微偏向通用化的产品形态，这类项目一般会强调自己为 CaaS（Container as a Service），例如：

1. 更多地对外暴露 Docker 容器各类概念；
2. 强调容器的 IP 而不是域名；
3. 不区分应用和它所依赖的“服务”；
4. 强调直接运维容器而不是应用。

这种类型的创业组织提供出来的更类似基于容器的 IaaS，意在给用户更大的自由度和发挥空间。当然，无论哪种形态，大家一般都会把 CI/CD 流程的支持纳入到自己的主要产品体系，毕竟这是 Docker 最受大家认同的一个场景；而且各家产品也都在功能上互相渗透，其实并没有一个非常明确的边界。从这个角度出发，当前的大多数创业组织的产品在大方向上会逐渐趋同，毕竟 Docker 本来的发展路线也是淡化云计算的分层理念，变轻，变薄。然后在小方向上营造差异化，比如有的组织会选择构建各类开发者工具形成产品链，有的会选择在 Infra 层面提供更优质、廉价、可靠的计算和存储资源（比如 SSD，高效的调度策略，最大程度压榨 IaaS 层利用率，甚至提供基于虚拟化技术的容器以彻底解决安全和隔离性问题）。总之，目前国内容器创业圈子产品能力优秀，相比之下即使是刚刚被 Docker 收购的 Tutum 恐怕在这一领域也没有太多优势；但是创新能力稍显不足，各个技术和产品点都是跟随者，尚没有展现出自己独有的优势。

2015 年国内容器技术圈的另一大事件便是巨头的跟进。各类互联网甚至传统技术企业在自己内部进行容器的规模化应用的案例早已不是新闻，而阿里云，阿里百川 TAE，新浪 SAE，网易等诸多厂商也都在 2015 年开始对外提供或基于容器提供云计算服务，我们相信还有更多的团队还在酝酿中。一般来说国内的 AE 类型的厂商（TAE SAE BAE）会优先选择提供 PaaS 类型的产品，原先的 IaaS 厂商则更愿意提供容器云服务。不管是哪种形态，国内容器服务的热度值的确做到了另前来布道的 Docker、Google 的老外们都惊叹不已的程度。但是稍微另笔者担忧的是，这种热度的发展，可能会过早的将国内容器技术提供商拖到价格

战的泥潭，届时大家过早停滞技术和产品的打磨而开始拼价格、拼渠道、拼运营，长远来看可能并不利于国内容器圈子的发展。

关于传统 PaaS 和 IaaS

国内容器技术热火朝天的背后，或多或少反衬出了传统 PaaS 和 IaaS 提供商的些许落寞。而事实上，传统 PaaS 和 IaaS 厂商在 2015 年依旧取得了不菲的成绩，仅以 Pivotal Cloud Foundry 为例，其商业产品已打入了大量国际一线制造业和金融业巨头，仅其中某一个大客户的日均运行容器数量恐怕目前尚没有哪家互联网公司能望其项背。就这一点而言，目前的 Docker 容器服务商恐怕在很长一段时间之后都很难出现这种规模和高质量的客户案例。但是商业归商业，开发者归开发者，Docker 为主的容器技术掀起的风潮起始于一线技术人员，也风靡于一线技术人员，这与商业产品的成功路线本质上就是不同的。这也正是为什么很多 PaaS 和 IaaS 厂商 2015 年甚至更早开始宣布支持 Docker 的最主要原因，OpenShift 甚至完全转型基于 Kubernetes 进行彻底重构。但是无论如何，在关系更密切的开发者服务场景下，目前 startup 做的更好。

另一方面，OpenStack 社区也在积极的寻求同 Docker 等容器技术的结合点来试图扩展一线技术人员这一不能忽视客户群，但是稍显遗憾的是哪怕是 2015 年被反复提及的 Magnum 项目都没有针对容器而对 OpenStack 本身做本质的改进和集成，项目依然是停留在调用 OpenStack API 然后把容器运行在 VM 里的程度。笔者认为，考虑到当前情况下虚拟机的不可替代性，OpenStack 如果能够提供虚拟机和物理机上容器的统一调度，或者引入类似 Hyper 或者 Clear Linux 这样的虚拟化技术容器，进而提供任务优先级、抢占、混部等一系列数据中心级别的高级特性，也不失为一种进取的手段。仅就 OpenStack 社区目前的应对方案来看，仍然不足以解决开发者的目光被 Docker 吸引并且采用容器作为 OpenStack 替代方案的情况。当然，OpenStack 无论是社区质量、规模还是产品、生态圈都不是现在的 Docker 所能挑战的，只是在社区层面对 Docker 以及容器技术的支持上做的还不够好。

总结

综上所述，2015 年的容器技术圈，是各家施展手脚封疆划土的扩张一年，也是 Docker 以及容器技术生态参与者不断完善自己的进化一年。总的来说，尽管有不少亮点涌现，这一年的容器技术仍然处于厚积阶段，标准尚未达成，社区缺乏平衡，跟进者多创新者少。但是，我们也不得不考虑到很可能这才是容器技术发展的一种常态，而不是重复其他开源项目的发展模式。或许在未来，容器技术生态圈的参与者始终会保持着这种不断进化的姿态以应对瞬息万变的应用场景和捉摸不定的开发者意图，很难达成一种平衡或者稳定的状态。毕竟在容器技术这种无限贴近于每一个一线技术人员的特殊领域里，“唯有适者，才能生存”。

作者简介：张磊，浙江大学 VLIS/SEL 实验室云计算团队负责人，Kubernetes 项目 Collaborator。

解读云计算：打磨产品服务，迎接市场变局

作者 魏星

在云计算领域，过去的这一年发生了一系列具有标志性的事件。公有云与私有云的争执尚未平息，容器云已经势如野火，开源技术生态圈的产品和服务仍需打磨，云安全将是企业长期关注的重点。而接下来的 2016 年，市场竞争将更为激烈，云厂商梯队间的距离将被拉大。

2015 年云计算行业标志性事件盘点

- 2015 年 4 月 1 日，OpenStack 项目创始者之一的 Nebula 公司宣布因市场原因而关闭。Nebula 成立于 2011 年，由 OpenStack 之父 Chris C. Kemp 创立。
- 2015 年 4 月 21 日，Citrix 宣布以企业赞助商的方式加入 OpenStack 基金会，不久后的 7 月，谷歌也加入了 OpenStack 基金会。2011 年

7 月，思杰收购了 Cloud.com，后者旗下拥有采用 GPLv3 授权协议的 CloudStack 开源项目。2012 年 4 月 5 日，思杰将 CloudStack 捐献给 Apache 基金会。

- 2015 年 4 月 27 日，杭州数梦工场和阿里云签署了战略合作协议，为客户提供专业的大数据和云计算服务。
- 2015 年 6 月 6 日下午，因服务商“睿江科技”机房遭遇雷暴天气引发电力故障，青云广东 1 区全部硬件设备意外关机重启，造成青云官网及控制台短时无法访问、部署于 GD1 的用户业务暂时不可用。
- 2015 年 7 月 30 日，华为在北京正式对外宣布“企业云”战略。在此前的 4 ～ 5 月间，华为要做公有云的传闻一度传得沸沸扬扬，国内的云厂商颇为惊讶。
- 2015 年 9 月 1 日，阿里云因云盾安骑士服务组件升级触发 bug，导致用户服务器可执行文件被误隔离。
- 2015 年 10 月 12 日，戴尔宣布将以约 670 亿美元收购数据存储厂商 EMC。戴尔公司创始人 Michael Dell 称：“Dell 与 EMC 合并后，新公司将成为一家企业解决方案巨头。”交易完成后，Michael 将出任合并后新公司的董事长兼 CEO。戴尔和 EMC 在联合声明中表示，交易完成后，VMware 将继续保持为一家独立公司。
- 2015 年 10 月 21 日，OpenStack 服务商 Mirantis 完成 1 亿美元的 C 轮融资，目前其总融资额已达 2.2 亿美元。Mirantis 是参与 OpenStack 项目的创始公司之一，作为 OpenStack 社区第三大代码贡献者，它开发出了一系列包括整合了不同功能的框架的工具库。
- 2015 年 10 月 22 日，惠普公司宣布退出 OpenStack 公有云市场。此前，惠普云服务主管 BillHilf 在接受《纽约时报》采访时表示“与亚马逊 AWS 等公有云服务巨头展开正面竞争对惠普没有意义”，这被解读为惠普正在放弃公有云业务，惠普迅速否认了这个说法。惠普从 2011 年开始在 OpenStack 方面投入研发，2012 年 OpenStack 基金会成立之初成为白金赞助商，并拥有两名董事会成员和五名技术委员会成员。2014 年 5 月，惠普基于 OpenStack 发布了 HP Helion。
- 2015 年 10 月 23 日，亚马逊公布了 AWS 业务 2015 年 Q3 财报，数据显示，AWS 净销售 20.85 亿美元，比去年同期的 11.69 亿美元增长 78%，在总

净销售额中所占比例为 8%；运营利润为 5.21 亿美元，高于去年同期的 9800 万美元，毛利润率从第二季度的 21.4% 增长到了 25%。

- 2015 年 11 月 1 日，在东京的 OpenStack 峰会上，UCloud 与 Mirantis 正式宣布成立合资公司 UMCloud，以求更好的在中国拓展市场。
- 2015 年 11 月 19 日，金山云正式对外发布三款企业级新产品，分别是云操作系统 KingStack，超融合存储 KingStore，和企业网盘服务 KingFile，在私有云市场开始发力。
- 在 2015 年 12 月的 ArchSummit 北京大会上，青云首次在公开演讲中透露了 QingCloud SDN/NFV 2.0 的技术细节。目前业内的 SDN 方案主要是通过硬件或软件两种方式来实现，而青云在设计第二代 SDN 方案时最终选择了与 NFV 进行结合。此外，青云也大力拓展其私有云市场。
- 2015 年 12 月 16 日，UnitedStack 有云宣布完成 C 轮融资，该轮融资由思科和红杉资本投资。UnitedStack 有云创始人 & CEO 程辉表示：“C 轮融资的完成，也是我们更快速发展的开始。未来，UnitedStack 有云将致力于为全球数据中心打造统一开放的云平台。”
- 2015 年 12 月，在国标委下达的 2015 年第三批国家标准修订计划中，正式下达 17 项云计算国家标准制修订计划。这是自 2013 年 10 月，我国正式启动可信云服务认证机制之后的一项重大政策调整。“可信云服务工作组”成员由工信部电信研究院、三家电信运营商、主要互联网企业和设备提供商组成。可信云服务认证的具体测评内容包括三大类共 16 项。

“公有云与私有云之争”

根据谷歌的分析，在未来的 2016 年里，34% 的企业将在接下来的两年中将超过 60% 的应用托管到云上，大部分供应商也会或已经采取措施支持企业业务云化的负载。此前，Gartner 也预测中国是仅次于美国和日本的第三大企业 IT 消费市场。2015 年，中国企业在技术产品和服务上的支出预计将达到 1464 亿美元，2016 年将增长 6.5%，达到 1558 亿美元，而到 2019 年，IaaS 的每年将增长 29.1%。

笔者参加 AWS re:Invent 2015 时感受颇深的一点是国内外云计算行业在市场方

面迥异。欧美市场公有云越来越火热，但中国市场并没有一家纯粹和专注的公有云提供商（阿里云可能算是唯一比较专注的），国内公有云的发展并不顺利，不少云厂商和业界同仁反复唱衰公有云和 OpenStack。随着 2015 年 Nebula 公司的关闭，以及美国几家 OpenStack 初创公司被低价并购，这一论调再次走高。中国有众多基于 OpenStack 发展业务的公司，包括几家用 OpenStack 做公有云的公司，能否在 OpenStack 界独树一帜、避免美国同行的命运，在公有云领域或者初创领域做出成绩？在接下来的 2016 年应该是一个分水岭。

公有云技术上最大的一个挑战就是成本。京东商城技术副总裁兼首席科学家何刚离开盛大的时候曾说过：

- 云计算没有过不去的技术门槛，更多决定于运营和维护。
- 云计算没有暴利，和游戏之类业务不同。云计算是堆机器，是拼服务。幻想暴利的可以绕道。
- 亚马逊采用 DevOps，一共 600 多人在 AWS 产品线，没有客服人员，用户自助、互助服务。但是国内做云计算会依赖客服。
- EC2、S3、EBS、RDS 是四个核心产品，其中都和 S3 有关系或依赖 S3。
- 网络是云计算最重要的基础和难点，同样网络部分也是 OpenStack 欠缺和急需改进的。

除了成本的挑战，另一个问题是用户使用习惯的挑战。国内的云计算用户基本都是 VMware 教育出来的。不少企业的 IT 技术尚处在虚拟化的阶段。因此，在谈用户需求的时候，国内的云服务提供商面临的往往是按照用户的需求来做。教育用户、引导用户是一件很痛苦的事情。蒋清野在谈国内公有云的发展时认为，公有云的成长要面临两个问题：一是用户增长，二是财务回报。而这两方面都不是目前国内公有云所能解决的。

客观地说，公有云用户的忠诚度较低——这个现象被一部分人当成了结果，造成的结果是国内云厂商把价格战当作主要运营手段。青云 CTO 甘泉在接受笔者的一次访谈时曾表示，这种把云资源当大白菜卖的做法更加剧了公有云的“赔钱赚吆喝”的现象，是一种不可取的运营策略。在一个充分竞争的市场中，用户忠诚度的问题通常是由产品同质化引起的。用户选择服务提供商，一是看重

其产品和服务的稳定、可用、可控，另一个则是价格因素。

云服务提供商首先要将其产品和服务设计成一个能赚钱的生意。而云服务中能赚钱的业务——网络和存储也不容乐观，SDN/NFV 技术目前尚不足以解决复杂的私有云问题，而 IDC、CDN 厂商的介入使得存储服务的钱也不那么好赚。

在现阶段的中国公有云市场中，用户还是把云服务当作 VPS 在用。蒋清野认为，究其根源，是公有云服务提供商所提供的服务也还局限在 VPS 上。这样一来，所谓的“公有云”只不过是传统数据中心或者是主机租赁服务的替代品了。一方面在于云厂商为了满足客户需求以签单而为之，另一方面在于云产品都围着 VPC 打转。但，即便是 VLAN 隔离也可以提供 4096 个逻辑上的隔离网络。除了阿里云之外，国内拥有了 4096 个隔离网络的企业级客户又有几多？换言之，为了 VPC 而 VPC 的做法有待商榷。

对于公有云的潜在超大规模的用户——互联网巨头（如 BAT、电商集团、在线视频、互联网金融以及在线教育企业，等）都选择自建云服务，这一点跟国外企业（典型的如 Netflix，等）选择 AWS、Azure、GCE 等大相径庭，不知道未来 IoT 的发展能否给国内公有云带来新的机会？

私有云的发展，品高软件副总经理周静在《2015 年政企私有云市场风云录》中多有阐述，在本文中就不再展开。即，私有云市场其实是传统的政企市场。这个领域原本都是经过外企教育多年市场——被规划好了基础架构、应用架构甚至 IT 管理架构，并培训了政企客户的 IT 管理员。因此，要认识企业云，必须认识到，云的构建是这一切重塑和调整的过程。2015 年被业界称为私有云的元年，此时中国的私有云进程跟全球站在近乎同一时间线上。2016 年这个市场必将爆发激烈的竞争。

OpenStack 与容器技术

著名博主、技术专家陈沙克日前撰文总结了 2015 年 OpenStack 技术的发展，他在文中提到：

1. 控制节点 HA：Mirantis 实现的控制节点的 HA 的方案，一直都是大家学习的对象，如果用户对控制节点进行严格测试，其实还是能发现有点美中不足的地方。
2. 计算节点 HA：对于 OpenStack 来说，确认一个节点挂掉，好像方法很多，监控，脚本，但是这些其实真正使用过程，你就会发现很多问题，还不如邮件通知，人工确认再迁移。
3. 增量快照和备份：目前默认 Openstack 把快照变成一个镜像模板的做法有些不妥。Ceph 跑分布式块存储，备份到 swift 的对象存储上的方案比较不错。如果 Ceph 备份到自己的对象存储上，风险还是没有降低。
4. 存储管理：Swift 已经发布了 5 年，其实功能本是足够稳定，不过你真的玩起来会发现，没有专人维护，根本不行。对存储系统的 Web 管理是刚需。
5. 监控和测试：Mirantis 是 OpenStack 的技术风向标，监控的数据的存储和展示，一直都是 ceilometer 社区折腾的问题。把目前 OpenStack 的各种方案都用好是一件非常有意义的事情。
6. SDN：Neutron+Vlan 的方案对于大部分企业来说，应该是基本够用，性能也都还可以。目前社区也就考虑逐步降低网络的复杂性，把文档里的 OVS 替换成 Linux Bridge。

早在 2014 年，OpenStack 社区就决定开始在技术上支持容器和第三方容器（如 Docker Swarm、Kubernetes、Mesos 等），并将其命名为“容器编制引擎（COEs，Container Orchestration Engines）”随着容器技术的持续火热，OpenStack 基金会在 2015 年下半年发布了自己的容器技术白皮书，开始支持 Linux 容器（LXC）和 Virtuozzo 系统容器。OpenStack 基金会首席运营官 Mark Collier 在 2015 年 5 月 OpenStack 峰会的主题报告中表示，就像 OpenStack 擅长帮助企业管理 VM 部署和虚拟化他们的数据中心一样，容器技术也可以做同样的事情。

2015 年，容器在生产环境中得到了广泛应用。最近的一项调查显示，2015 年容器应用增长了 5 倍。这部分得益于 Docker 和 Kubernetes 这类开源技术。国内容器技术进展迅速，互联网巨头甚至传统技术企业在自己内部进行容器的规模化应用的案例此起彼伏。如在大规模生产环境中实践的京东商城，以及提供

PaaS 服务的阿里百川 TAE、网易蜂巢等，我们相信还有更多的团队还在酝酿中。更多关于容器的盘点请参考《解读 2015 之容器篇》，本文不再赘述。

但容器在发展过程中颇受争议。10 年前，每一个虚拟机供应商都在吹捧自己的产品能够降低安全风险。但实际上非虚拟机系统所面对的安全问题虚拟机也有，时间证明黑客攻击并没有因为虚拟机而减少。对于容器也是这样，虽然其拥护者宣称应用和操作系统的相互分离能增强安全性，但事实并非如此，容器增加了复杂性，有与非容器系统一样的问题。另外，容器还增加了应用层的风险。

云安全

早年笔者采访黑客老鹰的时候他表示，云计算一方面让安全的防线拉得更长，另一方面让离散的风险变得更集中。动辄几百 G 的 DDoS 攻击，给企业也用户都带来了巨大的困扰，如果再加上不可抗力的雷电、不可预料的挖掘机以及如云盾安骑士升级这样的人为疏忽，云服务的安全的确令人难以相信。InfoWorld 安全领域的专栏作者 Roger A. Grimes 发表了一篇文章，总结了 2015 年的一些安全趋势，主要观点如下：

- 哪有什么匿名！
- 隐私比以往任何时候都重要。
- 物联网带来了更大的安全挑战。
- 容器加剧了安全问题的复杂性。
- TLS 取代了 SSL。

AWS 前 CISO Stephen Schmidt 在一次访谈中提到了 2015 年 AWS 在安全方面的建设工作。

1. AWS 高度关注加密机制，加密技术将覆盖到每一个产品领域。
2. 应对内部威胁的最佳途径就是限制指向数据的人为访问。
3. 对于第三方风险，通过审计确保各合作第三方与 AWS 遵循同样的安全标准。

加密技术和访问控制不但被 AWS 重视，Docker 也采用个这种安全策略。2015 年

11 月 16 ～ 17 日，在巴塞罗那举行的欧洲 DockerCon 上，Docker 宣布了一系列新的安全增强。这些增强功能包括容器镜像的硬件签名、镜像内容扫描和漏洞检测、用户细粒度的访问控制策略等。对于访问控制，前 Juniper 技术专家陈天在《深入了解 IAM 和访问控制》一文中有更多的介绍。

除了云平台自身健壮性的强化（如微软 Azure 正式启用了安全中心），国内外主要云平台都采用开放策略，通过开放 API 在自家的应用商店里集成更多专业的安全产品和服务。更多关于安全方面的解读，请关注『解读 2015』系列的安全篇。

小结

通过对 2015 年云计算行业的盘点，笔者总结如下，不足之处还请方家不吝赐教：

- 公有云服务竞争更趋激烈，国内外市场均出现洗牌迹象；国内云计算厂商开始进军海外市场。私有云市场需求大，但企业对使用云计算的信心尚有待提高。
- 技术方面，开源技术继续受到政、企和资本的追捧，围绕社区构建生态、打造产品和服务成为主流。新技术风潮走向更加务实的方向。
- 随着云计算产业规模的继续增长，明年的市场将出现分化，云厂商梯队之间差距会进一步拉大。

解读前端：工业时代野蛮发展

作者 刘奎

引用苏宁前端架构师徐飞的一个总结作为开篇。编程技术及生态发展的三个阶段：

1. 最初的时候人们忙着补全各种 API，代表着他们拥有的东西还很匮乏，需要在语言跟基础设施上继续完善；
2. 然后就开始各种模式，标志他们做的东西逐渐变大变复杂，需要更好的组织了；
3. 然后就是各类分层 MVC，MVP，MVVM 之类，可视化开发，自动化测试，团队协同系统等等，说明重视生产效率了，也就是所谓工程化。

处在 2015 年这个时间段来看，前端生态已经进入了第三阶段。看上去好像已经走的挺远了，实则不然。如果再用人类历史上的三次工业革命来类比，前端发展其实不过刚刚迈入了蒸汽机时代，开始逐步用工具来替代过往相当一部分的人肉作业，但是离电气时代的自动化流水线作业还有很长一段路要走。回顾一下 2015 年前端的生态发展，我大致整理了几个我觉得比较有历史意义的事件。

按时间顺序：

- 年初 React Native 的发布，引领 React 正式走上历史舞台；
- 3 月 angular2.0 第一个预览版发布；
- 5 月 http/2.0 标准正式发布，同月 iojs 与 nodejs 合并；
- 6 月 ES6 和 WebAssembly 落地；
- 7 月 迄今为止 React 生态圈影响最大的 Flux 实现 redux 发布 1.0 版本；
- 8 月 Facebook 公开了在 React 上应用 GraphQL 的 relay 框架的技术预览版；
- 9 月 React Native for Andriod 发布；
- 11 月伊始，ES 标准委员会宣布将历时 3 年研究的 Object.observe 从草案中移除，尽管它原本已经是 stage2，几乎已经是 ES7 的事实标准。双十一刚一结束，阿里手淘团队发布了名为 无线电商动态化解决方案的 Weex，也有人给了它一个更具象的名字，vue native；
- 12 月，赶在 2015 的尾巴，aurelia 和 angular2 先后发布 beta 版；
- css 方面，postcss & cssnext 先后高调走到台前。

观念的变化

由于近几年前端的野蛮生长以及前端应用的多元化和复杂化，整个技术形态已经跟几年前纯做页面的时代完全迥异了。主要观念的变化总结来看在于一点，现在的前端开发面向的是 web app 而不是 web page。今天的前端开发模式跟传统的 GUI 软件（如 C++、.NET 开发的 Windows 客户端）已经很接近了，而且由于现在前端领域为了解决日益复杂 的 web 业务需求及体量，越来越多的借鉴了传统客户端的开发经验，导致两者变得越来越趋同。再加上前端一些独特的特性（免安装、增量安装等），工程上的复杂度有过之而无不及。前端如今已经脱

离了茹毛饮血、刀耕火种的原始社会，开始步入了工业时代。

框架 & 类库的变化

今年最火的框架 / 类库毫无疑问当属 React 了。React 从 2014 年年中开始广泛受到开发者关注，但是真正开始在社区独领风骚还得归功于 2015 年初 React Native 的发布。React Native 的发布使得 js 统一三端（前端、后端、移动端）开发成为可能（现在这个时间点看可能还是过于理想，但是整体方向还是对的），这一针强心剂吸引了大量开发者的眼球。

我们挑几个主流的框架来讲讲这一层的变化。

React & Redux

React 可以参考这篇文章以及深入浅出 React 专栏，这里不再赘述。

Redux 则是目前 react 配套的 Flux 模式的各种实现中最火的一个，在此基础上它引入了函数式编程、单一数据源、不可变数据、中间件等概念。一定程度来讲，Redux 是今年 React 生态甚至整个前端生态中影响最大的一个框架，它给整个前端技术栈引入了很多新成员，尽管这些概念可能在其他领域已经有了广泛的应用。虽然它们是否会在大规模的应用实践中被广大开发者认可还需要再检验，但至少给我们带来了一些新的思路。

在我看来，React 的优势并不在组件化，React 的优势在于 virtual dom 及一个几乎构成闭环的强大生态，这归功于 Facebook 工程师强大的工程能力跟架构能力。virtual dom 将应用表现层从浏览器这个基于 dom 的上下文中抽离出来，通过原生 js 对象模型的方式使得 React 具备在任何环境支撑上层表现的能力。上层的渲染引擎可以是 canvas、native、服务端甚至是桌面端，只要相应的端提供基于 React 组件的渲染能力，即可达到一套代码、或者只要很少的改动就能

移植到任一终端环境的效果，这个就非常夸张了。React 从 0.14 版本之后便将 react-dom 抽出来变成一个独立的库，可见 React 的野心并不局限于浏览器，相反从这点来看，React 反而是受到了 dom 的掣肘。

Angular2 & Vue.js

Angular 2 因为不向下兼容引起了社区的争议，ng2 跟 ng1 相比是一个完全革命性版本而不是升级版，它是一个为了迎合未来的标准及理念来设计的全新框架，而这些新的理念又无法通过改进 ng1.x 的方式来实施，所以 Angular 团队做了这么一个看似激进的决策，可以理解成重构已经无法满足需求只能重写了。ng2 也采用纯组件化的开发思路，任何单元对于它来说都是组件。同时，ng2 里面也引入了一些全新的概念（对于前端而言）来提升框架的性能及设计，例如基于 worker 的数据检测机制能大幅度提升渲染性能（对应实现是 zone.js），基于响应式编程的新的编程模型能更大的改善编码体验（对应实现 RxJS）。赶在 2015 年的尾巴，ng2 正式发布 beta 版，对于 angular 的这次自我革命是否能成功，还有待后续检验。另外原 angular 团队 中出来的一个成员开发了一个类 ng2 的框架 aurelia，有相当的开发者认为它更配称为 ng2，值得关注。

由于阿里在背后的技术实践及支持，Vue.js 今年也开始得到越来越多的关注。vue 相对于 angular1.x 的优势在于轻量、易用、更优异的性能及面向组件化的设计，目前发展态势也非常好，是移动端开发的一个重要技术选型之一。

标准 & 语言的变化

现在回顾起来，2015 年是很有意义的一年：这一年是 Web 诞生 25 岁周年，也是 js 诞生的 20 周年。同时又是 ES6 标准落地的一年。ES6 是迄今为止 ECMAScript 标准最大的变革（如果不算上胎死腹中的 ES4 的话），带来了一系列令开发者兴奋的新特性。从目前 es 的进化速度来看，es 后面应该会变成一个个的 feature 发布而不是像以前那样大版本号的方式，所以现在官方也在推荐 ES+ 年份这种叫法而不是 ES+ 版本。

ES2015(ES6) & ES2016(ES7) & TypeScript

6 月中 ES2015 规范正式发布，从 ES2015 带来的这些革命性的新语法来看，JS 从此具备了用于开发大型应用的语言的基本要素：原生的 mudule 支持、原生的 class 关键字、更简洁的 api 及语法糖，更稳定的数据类型。而这些 new features 中，有几个我认为是会影响整个前端发展进程的：

- **Module & Module Loader**

  ES2015 中加入的原生模块机制支持可谓是意义最重大的 feature 了，且不说目前市面上五花八门的 module/loader 库，各种不同实现机制互不兼容也就罢了（其实这也是非常大的问题），关键是那些模块定义 / 装载语法都丑到爆炸，但是这也是无奈之举，在没有语言级别的支持下，js 只能做到这一步，正所谓巧妇难为无米之炊。ES2016 中的 Module 机制借鉴自 CommonJS，同时又提供了更优雅的关键字及语法（虽然也存在一些问题）。遗憾的是同样有重大价值的 Module Loader 在 2014 年底从 ES2015 草案中移除了，我猜测可能是对于浏览器而言 Module Loader 的支持遭遇了一些技术上的难点，从而暂时性的舍弃了这一 feature。但是一个原生支持的模块加载器是非常有意义的，相信它不久后还是会回归到 ES 规范中（目前由 WHATWG 组织在单独维护）。

- **Class**

  准确来说 class 关键字只是一个 js 里构造函数的语法糖而已，跟直接 function 写法无本质区别。只不过有了 Class 的原生支持后，js 的面向对象机制有了更多的可能性，比如衍生的 extends 关键字（虽然也只是语法糖）。

- **Promise & Reflect API**

  Promise 的诞生其实已经有几十年了，它被纳入 ES 规范最大意义在于，它将市面上各种异步实现库的最佳实践都标准化了。至于 Reflect API，它让 js 历史上第一次具备了元编程能力，这一特性足以让开发者们脑洞大开。

关于 ES2016 的最重磅的消息莫过于 11 月初 es 标准委员会宣布将 Object.observe 从 ES2016 草案中移除了，尽管它已经是 stage2 几乎已经是事实标准。官方给出的解释是，这 3 年的时间前端世界变化实在太大，社区已经有了一些更优秀简洁的实现了 (polymer 的 observe-js)，而且 React 带来的 immutable object 在社区的流行使得基于可变数据的 Object.observe 的处境变的尴尬，O.o 再继续下去的意义不大了。

除此之外，ES2016 的相关草案也已经确定了一大部分其他 new features。这里提两个我比较感兴趣的 new feature：

1. async/await：协程。ES2016 中 async/await 实际是对 Generator&Promise 的上层封装，几乎同步的写法写异步比 Promise 更优雅更简单，非常值得期待。
2. decorator：装饰器，其实等同于 Java 里面的注解。注解机制对于大型应用的开发的作用想必不用我过多赘述了。用过的同学都说好。

目前 ES2015/ES2016 都有了比较优秀的转译器支持（没错我说的是 babel），但是也不是 all features supported，尝新的过程中需要注意。

至于 Typescript，你可以将它理解成加入了静态类型的 js 的超集。这是完全开源运作的一个语言，其领导人为 C# 之父 Anders Hejlsberg，Angular 2 就使用它进行开发，感兴趣的同学可以了解一下。

WebAssembly

WebAssembly 选择了跟 ES2015 在同一天发布，其项目领头人是大名鼎鼎的 js 之父 Brendan Eich。WebAssembly 旨在解决 js 作为解释性语言的先天性能缺陷，试图通过在浏览器底层加入编译机制从而提高 js 性能。这个事情跟当时 V8 做的类似，V8 也因此一跃成为世界上跑的最快的 js 引擎。但是由于 js 是弱类型的动态语言，V8 很快就触碰到了性能优化的天花板，因为很多场景下还是免不了 recompile 的过程。因此 WebAssembly 索性将编译过程前移 (AOT)。WebAssembly 提供工具将各种语言转换成特定的字节码，浏览器直接面向字节

码编译程序。其实在此之前，firefox 已经搞过 asm.js 做类似的事情，只不过 WebAssembly 的方案更激进。有人认为 WebAssembly 可能是 2016 年最大的黑马，如果 WebAssembly 能发展起来，若干年后我们看 js 编写的应用会像现在看汇编语言写出的大型程序的感觉。WebAssembly 项目目前由苹果、谷歌、微软、Mozila 四大浏览器厂商共同推进，还是非常值得期待的。

Web Components

Web Components 规范起草于 2013 年，W3C 标准委员会意图提供一种浏览器级别的组件化解决方案，通过浏览器的原生支持定义一种标准化的组件开发方式。Web Components 提出之际引发了整个前端圈的躁动，大家似乎在跨框架的组件化方案上看到了曙光。但是前端这圈子发展实在太快了，在当前这个时间 点，Web Components 也遭遇到了跟 Object.observe 相似的尴尬处境。我们先来看看 webcomponents 的几个核心特性：

1. Shadow DOM
2. Custom Element
3. Template Element
4. HTML Imports

其中 1、4 现在都能很容易的通过自动化的工程手段解决了（shadow dom 对应的是 scoped css），而自定义标签这种事情不论是 React 还是 Angular 这类组件框架都能轻松解决，那么我用你 webcomponents 的理由呢？

另外 webcomponents 将目标对准的是 HTML 体系下的组件化，这一点跟 React 比就相对狭隘了（但是这并不表明 React 把战线拉的那么长就不会有问题）。

不过原生支持的跨框架的组件还是有存在的意义的，比如基础组件库，只是在当前来看 web components 发展还是有点营养不良。期待 2016 年能有实质上的突破吧。

架构的变化

2015 年出现的新的技术及思路，影响最大的就是技术选型及架构了。我们可以从下面几点来看看它对前端架构上都有哪些影响。

组件化

React 的风靡使得组件化的开发模式越来越被广大开发者关注。首先要肯定的是，组件化是一个非常值得去做的事情，它在工程上会大大提升项目的可维护性及拓展性，同时会带来一些代码可复用的附加效果。但这里要强调的一点是，组件化的指导策略一定是分治而不是复用，分治的目的是为了使得组件之间解耦跟正交，从而提高可维护性及多人协同开发效率。如果以复用为指导原则那么组件最后一定会发展到一个配置繁杂代码臃肿的状态。

工程化

工程化是近年前端提到最多的问题之一，而且个人认为是当前前端发展阶段最有价值的问题，也是前端开发通往工业化时代的必经之路。这里不赘述，有兴趣的同学看我前阵子整理的一篇文章前端工程化知识点回顾。

应用架构层 MVVM & Flux

MVVM 想必大部分前端都耳熟能详了，代表框架是 angular、vue、avalon。angular 在 1.2 版本之后加入了 controllerAs 语法，使得 controller 可以变成一个真正意义上的 VM，angular 整个架构也才真正能称之为严格的 MVVM(之前只能说是带有双向绑定的 MVC/MVP)。

Flux 是 facebook 随 React 一并推出的新的架构模型，核心概念是单向数据流。Flux 实质上就是一个演进版的中介者模式，不同的是它同时包装了 action、store、dispatcher、view 等概念。关于 Flux 对应用分层、数据在不同层之间只能单向流转的方式我是很赞成的。应用的分层在业务稍复杂的应用中都是很

有必要的，它更利于应用的伸缩及拓展，副作用是会带来一定的复杂度。

业务数据层 Relay & Falcor

这一层对大部分前端来说可能是比较新的概念，其实我们可以这样理解：在一个完整的应用中，业务数据层指的就是数据来源，在 angular 体系中可以等同于 ngResource 模块（准确来说应该是 $http）。

Relay 是 Facebook 推出的在 React 上应用 GraphQL 的框架，它的大致思路是：前端通过在应用中定义一系列的 schema 来声明需要的接口数据结构，后端配合 GraphQL 引擎返回相应的数据。整个事情对于前端来说意义简直是跨时代的，工业化典范！不仅能极大提升前后端协同的开发效率，还能增加前端对于应用完整的掌控力。但是目前来看问题就是实施过于复杂，而且还得后端服务支持，工程成本太高，这一点上 Meteor 显然做的更好。

Falcor 则是 Netflix 出品的一个数据拉取库，核心理念是单一数据源，跟 Redux 的单 store 概念一致。用法跟 Realy 类似，也需要前端定义数据 schema。另外还有一个新的 W3C 标准 api：fetch，它的级别等同于 XMLHttpRequest，旨在提供比 ajax 更优雅的资源获取方式，目前几个主流浏览器支持的都还不错，也有官方维护的 polyfill，几乎可以确定是未来的主流数据请求 api。

业务数据层是前端应用中比较新的概念，它的多元化主要会影响到应用的架构设计。

新的编程范式

函数式编程（FP）

函数式编程（functional programming）是近年比较火爆的一个编程范式，FP 基于 lambda 演算，与以图灵机为基础的指令式编程（Java、C++）有着明显的差异。

lambda 演算更关注输入输出，更符合自然行为场景，所以看上去更适合事件驱动的 web 体系，这点我也认同。但问题是，太多开发者看到 redux 那么火爆就急着学 redux 用 js 去玩函数式，我觉得这个有待商榷。如果你确实钟情于函数式，可以去玩玩那些更 functional 的语言 (Haskell、 Clojure 等)，而不是从 js 入手。最近看到一个老外关于 js 的函数式编程的看法，最后一句总结很精辟：Never forget that javascript hate you.

函数式响应型编程 (FRP)

函数式响应型编程 (functional reactive programming) 不是一个新概念，但也不过是近两年才引入到前端领域的，代表类库就是 ng2 在用的 rxjs。FRP 关注的是事件及对应的数据流， 你可以把它看作是一个基于事件总线 (event bus) 的观察者模式，它主要适用于以 GUI 为核心的交互软件中。但 FRP 最大的困难之处在于，如果你想使用这样的编程范式，那么你的整个系统必须以 reactive 为中心来规划。目前微软维护的 ReactiveX 项目已经有各种语言的实现版本，有兴趣的同学可以去了解下。

工具链的变化

去年最主流的前端构建工具还是 grunt&gulp，2015 年随着 react 的崛起和 web 标准的快速推进，一切又有了新的变化。

webpack & browserify & jspm

webpack 跟 browserify 本质上都是 module bundler，差异点在于 webpack 提供更强大的 loader 机制让其更变得更加灵活。当然，webpack 的流行自然还是离不开背后的 react 跟 facebook。但是从现在 HTTP/2 标准的应用及实施进展来看，webpack/browserify 这种基于 bundle 的打包工具也面临着被 历史车轮碾过的危机，相对的基于 module loader 的 jspm 反而更具前景。

PostCSS & cssnext

PostCSS 作为新一代的 css 处理器大有取 Sass/Less 而代之的趋势，Bootstrap v5 也有着基于 PostCSS 去开发的计划。但从本质来讲它又不算一个处理器，它更像是一个插件平台，能通过配置各种插件从而实现预处理器跟后处理器的效果。

cssnext 官方口号是“使用来自未来的语法开发 css，就在今天！”，但是 cssnext 又不是 css4，它是一个能让开发者现在就享受最新的 css 语法（包括自定义属性、css 变量等）的转换工具。

写在最后

从前端的发展现状来看，未来理想的前端技术架构应该是每一层都是可组装的，框架这种重型组合的适用场景会越来越局限。原因在于各部件不可拆卸会增加架构的升级代价同时会限制应用的灵活性。举个例子，我有一套面向 pc 端的后台管控平台的架构，view 层采用 angular 开发，哪天我要迁移到移动端来，angular 性能不行啊，我换成 vue 就好了。哪天觉得 ajax 的写法太挫，把 http 层替换成 fetch 就好了。又有一天后端的 GranphQL 平台搭好了，我把 ngResource 换成 relay 就 OK 了。

这种理想的方式当然是完全正确的方向，但是目前来看它对开发者 / 架构师的要求还是太高，工业级别上一套带有约束性的框架还是有相当的需求的。虽然美好但是组合的方式也不是没有问题，各种五花八门的搭配容易造成社区的分化跟内耗，一定程度上不利于整个生态圈的发展。

近年前端生态的野蛮发展影响最大的应该就是新产品的技术选型了，乱花迷人眼，我们很难设计出一套适应大部分场景、而且短时间内不会被淘汰的架构。前端的变化太快通常会导致一些技术决策的反复，今天的最佳实践很可能明天就被视为反模式。难道最合适的态度是各种保留各种观望，以不变应万变？那句话怎么说的来着？从来没有哪个圈子像今天的前端一样混乱又欣欣向荣了。

有人说 2015 年或许是大前端时代的元年，目前看来，如果不是 2015，那么它也一定会是 2016 年。

最后引用计子 winter 的一句话作为结语吧：

前端一直是一个变化很快的职能，它太年轻，年轻意味着可能性和机会，也意味着不成熟和痛苦。我经常担心的事情就是，很可能走到最后，我们会发现，我们做了很多，却还是一无所获。所幸至今回顾，每年还是总有点不同，也算给行业贡献了些经验值吧。

本文作者刘奎（@kuitos），原文地址，本文由作者授权转载，相对原文有删减。

作者介绍：刘奎，一个后端出身的伪前端工程师，如今专注于前端领域。目前在国内一家从事电商 CRM 服务的互联网公司供职，主要负责公司各产品的前端架构、技术选型及前端体系化搭建。涉猎的领域包括 SPA 应用架构、前端工程化、web 标准等等。中度代码洁癖症患者，唯标准论的教条主义者。

解读安全：安全环境持续恶化

作者 乌云君

2015 年 12 月 16 日，在乌云 TangScan 的发布会上，我跟邬迪说这一年安全领域变化很大，各路安全峰会从年初开到年尾，安全理念也从“气宗与剑宗”发展到了“南向与北向”，作为第三方平台乌云理应盘点一下 2015 年的变化。近日报告初成，可以看到过去的这一年在安全领域，安全事件有增无减；数据隐私泄漏更为严重；第三方平台开始出现严重的信任危急；企业员工安全意识薄弱令人触目惊心；而智能设备的不断涌现，万物互联使得本就脆弱的基础设施安全更加千疮百孔。

对于 IT 从业者

热点漏洞

0x00 打印机居然接入互联网

4 月 29 日，乌云平台报告了中国移动某省公司打印机未授权访问的漏洞。由于

该公司网络部打印机接入互联网可直接外网访问，导致黑客可以通过该打印机下载曾经打印过的文档。作为企业来说，对于需要联网的办公设备，应该尽可能保证不对外开放，毕竟一切接入互联网的东西都是存在安全风险的，何况是存储着企业资料的打印机呢？

贷齐乐借贷系统出现多处注入漏洞，影响上百家 P2P 平台

8 月 8 日，乌云匿名白帽子路人甲报告了知名 P2P 系统贷齐乐存在多处 SQL 注入漏洞，可影响大量 P2P 网贷站点，并随手附上了 100+ 案例。据了解，贷齐乐公司是一家专业从事 P2P 网络借贷平台开发的战略性电子商务有限公司，目前全国百分之七十以上的借贷网站都是用贷齐乐系统搭建，当贷齐乐系统出现多处注入之后，上百家 P2P 平台都受到了漏洞影响。

Java 反序列化漏洞爆发，风头直逼当年的 Struts 2

11 月 6 日，国外 FoxGlove 安全研究团队在其博客上公开了一篇关于常见 Java 应用如何利用反序列化操作进行远程命令执行的文章，其中详细说明了 Java 中如何使用 Apache Commons Collections 这个常用库来构造 POP 链（类 ROP 链）来进行任意命令执行。从 11 月份 Java 反序列化漏洞爆发开始，2015 年的最后两个月里，乌云平台已经陆续收到白帽子报告的此类漏洞数量多达 900+，这个数字还在继续增加，影响范围直逼当年的 Struts 2 漏洞。

Redis 后门植入

11 月 10 日，国外安全研究者的一份文档显示，Redis 在未进行有效验证，并且服务器对外开启了 SSH 服务的前提下，攻击者有可能恶意登录服务器甚至进行提权操作（root 权限）。通过与一些企业和机构的沟通，已经发现了大量扫描与自动化攻击痕迹。

乌云君说：从 2010 年乌云平台创立至今，已不断收到了包括 Struts、Jboss、Weblogic、ElasticSearch、Redis、Java、短信平台等第三方软件或服务的相关漏洞，这些漏洞已影响了无数厂商。随着使用者数量的增多，第三方软件或服务的漏洞影响范围还在不断扩大。对于使用第三方软件或服务的企业，需要密切关注漏洞预警，及时打上相关补丁，做好防护工作，降低损失。

0x02 XCodeGhost 引发的信任危机

9 月 14 日，一例 XCode 非官方版本恶意代码污染事件逐步被关注，并成为社会热点事件。多数分析者将这一事件称为“XCodeGhost”。截止到 9 月 20 日就已确认共 692 种（如按版本号计算为 858 个）APP 曾受到污染，受影响的厂商中包括了微信、滴滴、网易云音乐等著名应用。（我能说“珍爱生命、远离国产”吗？）

乌云君说：此次受到污染的 XCode 虽然是非官方渠道下载的，但开发者们却不约而同的默认了它的安全性与正规性，这也可以看出企业在开发流程方面的不规范。经由此次事件可以看出，开发流程的规范是必不可少的，开发直接接触企业与用户之间沟通的产品，用户体验不只是使用时的舒适感，更是使用时的安全和放心。

0x03 WormHole：比葫芦娃还可怕的百度“全家桶”漏洞

7 月 28 日，乌云收到了第一例“虫洞”漏洞报告，白帽子发现百度助手开启的某个奇怪的数据通讯端口提供了一系列的信息查询功能，分析后猜测这个功能应该是用作与百度统计进行一些用户信息的交互回调。同网络环境中的其他用户可以直接访问这个端口并获取用户敏感信息，后来白帽子又陆续有了新的发现，比如新的监听端口以及接口功能等。该漏洞属于通用性质，国内其他厂商 APP、SDK 也可能存在类似问题。

乌云君说：苹果 CEO 库克同志曾经说过，“你不应该给软件装后门因为你不能保证这个后门只有好人能够使用”。不论厂商由于何种缘故而在软件中添加后门，都是一种不可取的做法，一旦后门被不法分子利用，造成的后果不仅是让用户的敏感信息不保，更会伤害用户对于企业的信任，这实在是种得不偿失的做法。

0x04 当企业对自己所处领域的安全技术信息不对称时

9 月 6 日，乌云收到了白帽子提交的有关小米手环的漏洞，该漏洞可以使黑客通过使用 Lightblue 来控制手环狂震不止。漏洞是由于蓝牙可以用 BLE 读写工具在不匹配的过程中直接读写设备数据造成的。

乌云君说：在智能硬件这个新兴行业中，企业还处于对该行业技术的摸索与上

升阶段，目前大多智能硬件都使用蓝牙作为设备连接的方式，不可否认这也是一种非常合适的技术。不过现在大部分设备都还停留在无认证无加密的链路默认模式，这是非常不靠谱的。

这个漏洞从侧面反映了智能硬件行业中，企业对于所使用的技术的安全性信息不对称，场景考虑不够全面。虽然企业可能有关于产品开发周期与成本的顾虑，但还是缺少了对于一些场景的思考，尽管智能硬件行业的攻防现在还属于很小众的话题，硬件的开发和存储空间对于开发也是一个挑战，但对于一家想要用心做好产品的企业来说，对于安全的敏感性和掌控性仍旧不可以丢失。

热点事件

0x00 企业安全中最脆弱的一环——人

9 月 25 日，苏宁易购 iOS 源代码被发现出现在 GitHub 上，导致大量敏感信息泄漏。这已经不是第一起“人为”的安全事故了。1 月 6 日，乌云平台发现大量将各类帐号密码公布在公告中的 QQ 群；6 月 13 日，白帽子提交的编号为 WooYun-2015-119843 的漏洞就是因为中国联通工号申请审核人员的安全意识薄弱而给了黑客可乘之机……一连串人为造成的安全隐患证明了每一个不起眼的个体都能成为安全环节中的致命点，企业安全中「全员」的安全意识也应成为重中之重。

0x01 被盯上的办公网环境

4 月 16 日白帽子提交了一个通过破解京东某办公分部的 WiFi 而成功漫游京东内网的漏洞（WooYun-2015-108465）。其实因为办公网 WiFi 密码泄漏而造成的黑客侵入内网事件不单单出现在京东，早在 2 月 14 日，就有白帽子连接“天河一号”某用户的企业办公 WLAN 后导致白帽子漫游“天河一号”；10 月 27 日，小米科技因为售后授权中心的办公网 wifi 密码被破解而使得白帽子成功进入售后系统……企业中很多重要设施或系统因为安全起见都被放在内网环境，但是当办公网密码被破解，内网还能安全吗？

0x02 企业员工的日常安全

1 月 14 日，TCL 某网站在迁移代码时把 Mac 的隐藏文件 DS _Store 搬了过去，

导致目录结构泄漏，从而可以使黑客获取到工具扫不出来的后台管理页面以及数据库文件。现在大部分企业在开发架构时会选择使用 Mac，但 Mac 具有系统特殊性，在编辑文件时会自动生成隐藏文件 DS _Store ，该文件会包含很多网站敏感信息，成为网站安全性的致命点。安全在于细节，细节决定成败。

0x03 企业安全机制的根本设计导致安全问题

1 月 15 日，酷派由于短信服务配置不当导致酷派云所有用户信息泄漏，酷派云注册找回密码时的短信网关因为没有任何限制而可以直接访问，其中包含大量用户敏感信息。本来作为安全机制使用的“找回密码”由于根本设计上的风险而成为安全中的最大障碍，这总让人有种无限惋惜和难以置信的感觉。安全不仅要注意每一处微小的细节，更要从根本上就规避可能隐藏的安全风险，安全真的容不得一丝疏漏，谁知道哪处疏漏就能造成怎样的危害呢？

对于白帽子

热点漏洞

0x00 一切输入皆有风险

1 月 18 日乌云公开了一个通过微信公众号获取到了大量优酷内部员工敏感信息，并存在定向钓鱼风险的漏洞。由于后端开发时什么都没有过滤，所以只需要往优酷内部的某公众号发送注入语句，就能获取到大量内部用户信息，甚至还可以把数据 Dump 下来，让我们来幻想一下：如果把所有用户数据都拖了出来，然后解出几个密码，顺利登陆邮箱 VPN 啥的，会出现什么后果呢？另外，通过该公众号，白帽子还找到了该公众号的微信后台管理入口并利用弱口令成功登录，这一点可以被用来做什么样的钓鱼我们就不多说了。虽然是面向内部员工开放的公众号，但既然是公众号，那就要做好会被“公众”知晓的准备，后端开发怎么可以如此马虎呢？要记得：一切输入皆有风险。

0x01 万物互联之车联网的安全风险

听说我们现在正在逐渐步入「万物互联」的物联网时代，这其中会存在怎样的安全风险呢？ 3 月 30 日，乌云“著名白帽子”路人甲报告了一例有关比亚迪智

能汽车的漏洞（WooYun-2015-104734），车联网也开始成为了黑客的攻击目标。该漏洞是由于在比亚迪云服务页面，浏览器发送 AJAX 请求判断手机号存在时，会返回车主包括姓名、车牌号、车架号、身份证号等个人敏感信息，利用此漏洞，可以遍历手机号段来获取车主信息和控制密码。这是最好的时代，也是最坏的时代，「万物互联」带来的不只是生活上的便利，似乎，还有些别的东西，你觉得呢？

0x02 安全设备真的可以成为我们的“保护神”吗

你们买安全设备是为了什么？当然这是一个很傻的问题——买安全设备当然是为了保证安全啊！可是你有想过有一天你买的安全设备也可能带来安全问题吗？4 月 2 日，有白帽子提交了一个天融信应用交付系统源码泄漏的漏洞（WooYun-2015-105415），并且使用 CloudEye 直接黑盒测试，发现可以存在命令执行。在这个漏洞中，源码泄漏的方式非常有意思，就是在 php 页面的 url 后加了个「.」，就成功获取到了源代码，这让人有些哭笑不得。安全设备出现安全问题会让用户的第一道防线出现破绽，而其中封装的黑盒也会隐藏着很多难以察觉的不确定因素，更需认真对待。

0x03 可以被任意下载的离线文件

漏洞的影响范围往往取决于用户量的多少。9 月 25 日，腾讯 QQ 被报存在高危漏洞可以读取并下载任意用户离线文件，导致用户敏感信息泄漏（WooYun-2015-143395）。这个漏洞的实现手法非常简单，但影响范围还是比较可观的，毕竟 QQ 作为目前国人主要的社交聊天软件，用户群还是非常庞大的。那么问题来了，当你承载着那么庞大用户量的各类隐私时，怎么可以不做好服务端校验等安全小细节呢？

热点事件

0x00 Hacking Team 的“军火库”泄漏

7 月份，意大利一家专业向政府及执法机构贩售入侵与监视工具的意大利黑客公司 Hacking Team 被攻击，攻击者对公众发布了多达 400GB 的内部文件、源代码以及电子邮件供任意下载。攻击者黑掉了 Hacking Team 的 Twitter 账户，丑化

了 Logo、简介，并且将获得的内部消息通过该公司的 Twitter 公布于众。

对此，乌云君只想说，夜路走多了总会遇见鬼的，声名狼藉的 Hacking Team 也会被黑。技术应该被用在正确的道路上，尊重民众的隐私，保护个人信息安全才是安全从业者们想要做和应该做的。

0x01 “无敌舰队”的 DDoS 勒索

11 月份，国外一支名为“Armada Collective”的黑客团队（即“无敌舰队”）利用 DDoS 攻击技术开始勒索企业，其中有很多值得国人警惕的地方。该团伙作案都是瞄准一些大目标，先发邮件勒索，然后会随时发起 1Tbps 流量的 DDoS 攻击。攻击先从国外开始，但从“无敌舰队”后来的邮件中可以看出他们已经对中国国内做了踩点分析。一直都说国外互联网目前遇到的问题，就是国内互联网未来要面对的。目前来看这进度快同步了，至少是互联网的阴暗面——网络勒索攻击已经开始试运行了。

对于普通用户

热点漏洞

0x00 智能硬件安全标准贫瘠

WooYun-2015-134839、WooYun-2015-143270、WooYun-2015-143181…

2015 年 9 月，乌云平台上一支白帽团队研究了市场上十余款畅销智能儿童手表，发现大半存在漏洞可远程定位、窃听孩子并匹配身份，甚至还能切断手表和父母手机的联系。为保障孩子安全而买的儿童手表，因其简陋的安全防范却变身成了不怀好意者的窃听利器。谁来监督智能硬件的安全性？

0x01 云端投毒

WooYun-2015-128592、WooYun-2015-139347

2015 年 7 月、9 月，乌云平台连续收到两个云端风险案例。7 月有白帽发现某广告屏蔽软件云端的屏蔽规则审核后台存在漏洞，可获得管理员权限，这款软件有数十万用户，任意一个规则出错都影响巨大；9 月另一白帽发现，某智能路由固件存储了一个远程服务器认证信息，登录后发现为该品牌所有路由的云端管

理调试平台，可 ROOT 权限远程控制任意路由。当“云”成为趋势，厂商如何保障它的安全？

0x02 越狱插件窃密

WooYun-2015-136806

似乎只有惨痛教训，才能让大众对安全有所重视。2015 月 8 月，有白帽向乌云平台报告，发现多款越狱软件内置后门，会窃取用户 iCloud 账号密码及机密信息。白帽从窃密软件后台找到了 22 万 iCloud 账号密码，抽样测试发现多为有效账号，可直接登录。未知来源的软件可能不安全！再强调一百年也不为过。

热点事件

0x00 大麦 && 网易帐号泄漏事件

2015 年 8 月 26 日，乌云报告平台显示，大麦网再次被发现存在安全漏洞，600 余万用户账户密码遭到泄漏，有白帽子甚至发现，这些隐私数据已被黑产行业进行售卖与传播；10 月份，有网易邮箱用户在论坛和微博上反映自己的网易邮箱泄漏，一些用网易邮箱注册的第三方账户被盗。

2015 年的数据泄漏事件不仅让大众更加明白了数据的重要性，提高了对于数据的安全意识，也让民众知道了「脱库」和「撞库」两个安全名词。

「脱库」和「撞库」是具有本质区别的，前者是黑客具有主动权，泄漏的数据也更完整，危害当然也更大；后者黑客是被动的，能否撞到数据全凭运气以及网民的安全意识，危害当然相对而言较小。

小结

如果说 2014 年是史诗级漏洞频发的一年，2015 年则是安全环境持续恶化的一年。我们看到广大企业并没有平衡业务与安全的关系，甚至为了业务而牺牲安全。尽管一再有血的教训，但麻木的大众对频发的安全事件越来越冷漠，IT 从业者的安全意识依旧令人担忧。在接下来的一年里，这一现状能否有所改善，恐怕还需要企业、白帽子以及社会大众的共同努力。

解读运维：变化、发展、涨姿势

作者 刘宇

本文为“解读 2015 之运维篇”。2015 年虽不是运维发展最快的一年，却是运维变化最大的一年。前一年大家普遍认为云计算是运维的救命稻草，而 2015 年的多次“灾难”让人警醒。越来越多的企业开始对运维更加重视了，大力提倡自动化运维，围绕 “自动化运维”相关的探讨也越来越频繁，在一次又一次的思想碰撞中擦出火花，并实施落地。与此同时，无论是使用开源软件的数量还是采用开源软件的企业都在持续增长。许多公司对自己的一些软件进行了开源，其中包括 Google、Facebook、微软和 IBM 等。企业用户以前所未有的速度拥抱开源，很多优秀人才也投身其中，开源不再可有可无。除此之外，运维基础系统的更新迭代也不容小觑。还有哪些涨姿势？我们一起来盘点。

运维的多事之秋

[2015.5.27] 支付宝因杭州机房网络光纤被挖，导致数小时部分用户业务不可用

[2015.5.28] 携程网瘫痪事件，全网业务中断 12 小时
[2015.6.1]UPYUN 连续遭遇两次大规模流量攻击，影响业务 6 小时
[2015.6.6]QingCloud 因雷暴引起的广东 1 区 IDC 电力故障，业务中断 2 小时
[2015.6.6]LeanCloud 多项服务发生中断，持续 4 小时
[2015.6.15] 知乎机房故障，影响系统使用近 2 小时
[2015.6.21] 阿里云香港节点宕机，业务中断 13 小时
[2015.6.19] 开源中国 Git@OSC 连续遭受 DDoS 攻击
[2015.09.01] 阿里云升级云盾引入 BUG，导致误删用户文件
[2015.09.22] 七牛云存储服务故障，业务中断 83 分钟
……

通过这些不完全统计，可以看出云故障是比较多的，在出现故障后，虽然公司会有财务及形象上的损失，但是心态一定要好，不能手忙脚乱。沉着应对，高效处理，快速恢复才是运维人员的“正确姿势”。2015 年，我们从这些故障中不断地总结，在技术提升的同时，更应该转变的是观念，正视容灾备份的重要性，将风险降到最低。伴随着基础设施的逐步完善，未来这些层面都不再是问题，再者也可以利用工具的优势来解决高用性架构。

运维工具组合的进化

随着云计算和开源的高速发展，大量应用需要横跨不同网络终端，并广泛接入第三方服务，IT 系统架构越来越复杂。快速迭代的产品需求和良好的用户体验，需要运维管理者时刻保障核心业务稳定可用，企业运维中的痛点和难点也急需解决。以下运维工具在这一年更加火爆，为企业业务提供强有力支撑：

- 命令执行与配置管理
  - Ansible
  - SaltStack
  - Puppet
- 持续交付与代码
  - Jenkins
  - 国内 Coding.net, GitCafe，Git@OSC 的兴起

    - GitLab 的进步与稳定
- ELK 生态的成熟
    - 提供日志收集，分析，和实时搜索，与可视化监控
    - 最近发布 2.0 大版本
- 应用监控
    - APM
- 国内开源
    - open-falcon

一套好的运维工具，能够将应用、网络、计算、存储、虚拟化等资源的性能及告警信息综合分析（可视化），通过简洁易懂的界面，直观呈现业务健康水平。当出现故障时，能够先从全部业务的宏观视角，确定关联和影响，再通过智能钻取和故障定位技术，缩小故障定位范围是在计算、应用还是网络，从而明确问题职责，帮助运维和研发准人员确定位业务故障位置。国内的运维人员所需要做的就是利用 Django 结合开源工具，开发出适合自己业务系统的平台，通过一定的流程控制，将业务紧密贴合，从而逐步达到自动化运维的目的。

运维基础系统的进化

企业最重要的是业务系统，所有的工作都是围绕正常开展业务而展开的。而 Linux 系统更是技术人员最基本的基石。今年 4 月份 Linux4.0 发布，这是一个新的里程碑，更新“live patching”（实时补丁）机制，意味着以后为内核打补丁不用重启系统了，增强了系统的高可用性。

随着容器技术的兴起，许多新的专门运行容器的 Linux 发行版本也出现了。光版本就迭代了 9 个，火得不要不要的。这应该是其它开源软件所无法超越的，同时也整合了工具集合：Toolbox，为部署提供便利。不过这也使得原本为 Docker 做出巨大贡献的 CoreOS 与 Docker 分道扬镳，并独立发布了 Rocket(rkt)，认为 Docker 已经忘记初心，从而独立出来做一个更纯净的容器。Rocket 没有像 Docker 那些为企业用户提供的“友好功能”，比如云服务加速工具、集群系统等。反过来说，Rocket 想做的，是一个更纯粹的业界标准。

运维脚本语言的进化

提到脚本语言，今年最火的非 Python 莫属，同时也是 Python 社区稳定改善的一年。这一年 Python2 由 2.7.9 进化到 2.7.11，然而 2.7 版本可以说是非常稳定成熟，用 Python 之父 Guido 的话说：“是想不到任何可以加入的新特性，因此不会发布 2.8 版本”。同时大量重要开源库通过 six 兼容包来同时支持 Python2 和 3，这让开发者转型 Python3 变得不再那么遥不可及。Python3 在 9 月份横空出世了 3.5 版本，正式宣告 Python 成为一个从语法上原生支持协程的语言，这一特性也吸引着越来越多的开发者迁移到 3.5，越来越多的开源库迁移到 Python，这个方向的改变是非常明显的。

除了 Python 外，Django 无疑也是一匹黑马，运维本来跟 Web 开发没有太多关联，但 Django 的长足进展，却帮了国内运维一个大忙。在大力提倡运维开发的时代，Django 的出现让运维非常快速方便地开发部署自动化工具，极大地释放了运维的生产力。2015 年，Django 从 1.7 升级到 1.9，支持大量新特性，开发部署越来越简便，而且生态越来越成熟， 2016 年即将朝着 Django2.0 的方向迈进。

总结

这一年，在大规模、复杂架构的催生下，运维技术不断变化、发展、涨姿势。自动化运维被推到一个新的高度，给传统企业带来了福音；给基础运维带来了巨大的挑战与机遇；同时也给越来越多的企业带来了新的抉择；开源技术的飞跃、脚本语言的进化等也给运维行业带来了革命性的影响。展望 2016 年，相信容器技术将持续爆炸式增长，云运维更加简单高效，让我们拭目以待，尽情拥抱他们吧。

免费在线版本

（非印刷免费在线版）

InfoQ 中文站出品

本书由 InfoQ 中文站免费发放，如果您从其他渠道获取本书，请注册 InfoQ 中文站以支持作者和出版商，并免费下载更多 InfoQ 企业软件开发系列图书。

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