适合开发者 项目经理 CTO&CIO 编程爱好者阅读收藏

Broadview®
www.broadview.com.cn

PROGRAMMER 程序员

PROGRAMMER 程序员 2015 精华本

程序员编辑部 编

7大篇章，24期精华 讲述成功产品背后的 **技术、人和事**

人物篇 ■ 专题篇 ■ 云计算与大数据篇 ■ 移动篇 ■ 技术篇 ■ 管理篇

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

程序员网址：http://programmer.csdn.net

程序员2015精华本

程序员编辑部　编

总策划：孟迎霞

编委会：蒋　涛　钱曙光　仲　浩　卢　凯　唐小引

魏　伟　陈秋歌　周建丁　纪明超

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

程序员 2015 精华本

图书在版编目（CIP）数据

程序员 2015 精华本 /程序员编辑部编. —北京：电子工业出版社，2016.3
ISBN 978-7-121-28295-9

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

中国版本图书馆 CIP 数据核字（2016）第 047621 号

责任编辑：董　英
印　　刷：北京京科印刷有限公司
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：850×1168　1/16　印张：38　字数：1672 千字
版　　次：2016 年 3 月第 1 版
印　　次：2016 年 3 月第 1 次印刷
定　　价：79.00元

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

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

服务热线：（010）88258888。

前　言

效法工业

有人曾问法国印象画派先驱 Agugste Renoir，为何他总是露天作画，而非于画室中创作？Renoir 回答说，在画室中，他大概只能想象出四五种不同模样的树叶，再画其他叶子便与之相似。而大自然则创造出了百万种不同的树木，是取之不尽的思维源泉。Leslie Lamport（图灵奖得主，LaTeX 设计者）则说，正是这样的原因，他从未供职于大学，而是始终在工业界开展他的研究。《程序员精华本》中的每篇技术文章都来自生产环境中遇到的真实问题，由当事人为你分析案，阐释原因。

假如你入行不久，这本书将让你身临其境，感受行业中最真实的一面——紧迫的问题从何而来，缘何非克服不可。如果你已是摸爬滚打多年的老手，这本书不但能让你学到有效具体的技术方案，还能重现资深从业者的思考方式——让你领略如果站在作者的角度上，会如何思考，怎样判断。

你需要的最新资料

Marvin Minsky 曾开玩笑说，每样科学都有“半衰期”，而计算机科学可能是半衰期最短的领域之一——洗个澡的工夫，就有一半知识要过时了。

在你离开学校前，有的知识就已多年未曾更新（比如新标准下编程语言的用法），而坊间口口相传的，或许只是“过时”技术在各自头脑中酝酿的不同版本——或许你并不知道，走在行业前沿的开发者，早已拥有了更好的方案，而用户最关心的问题也已经默默发生着变化。《程序员精华本》为你收录最近一年的行业进展，在文章重新组合的过程中，我们保留了那些让你能在今天受益的文章——向你讲述新标准、新工具、新解法、新问题，以及新的思考方式。

带给你意外收获

1986 年，科学家 Richard W. Hamming 在贝尔通讯研究中心，为两百名科学家做了题为《You and Your Re-search》的演讲：

“我观察到，关上办公室的门，确实能在今天明天完成更多工作，效率也比其他人高。而十年后则未必——你已分不清哪些问题才真有价值了。而敞开门的人，总免不了各种打扰，却在偶然间获得了有关这个世界是什么，以及哪些问题更重要的线索。敞开门与最终成就大事之间，联系千丝万缕，尽管关着门的人通常更用功”。

重要的发现常常来自“意外收获”——科学家依靠同事带给他们洞见。而阅读《程序员精华本》，将让你获得“这个世界是什么，以及哪些问题更重要”的线索。这本书不仅有你最迫切想要了解的问题，也会告诉你行业中还有哪些问题很重要。这些线索会不会帮助你翻开职业生涯新的一页？我们拭目以待。

程序员编辑部

目 录

人物篇

专题篇

云计算与大数据篇

移动篇

技术篇

管理篇

“新”科学家 Stephen Wolfram

记者 / 卢鹄翔

Stephen Wolfram 1959 年出生于伦敦，他几乎获得了所有少年天才能拥有的成就——13 岁进入伊顿公学，15 岁发表首篇物理学论文。两年后，在牛津大学一年级暑假，大部分学生都会打工挣钱或者搭顺风车周游欧洲，Wolfram 却写了一篇关于“量子色动力学”的论文，诺贝尔物理学奖得主 Murray Gell-Mann 注意到这篇论文，邀请 Wolfram 加入他在加州理工学院的研究小组。两年后，Wolfram 获得了理论物理博士学位，这时他才 20 岁（他的博士导师之一 Hugh Politzer 日后也获得了诺贝尔物理学奖）。他留在加州理工学院任教，并与另一位诺贝尔奖得主 Richard Feynman 共事。此后不久，Wolfram 获得了麦克阿瑟天才奖，是该奖项最年轻的获得者之一。

物理学家的计算机之路

“我从来都不喜欢算数字，不过大约从 10 岁起，我对物理开始着了迷——而研究物理离不开数学。” Wolfram 回忆说，他 12 岁那年第一次接触了电子计算机——书桌般大小，有 18 位 8 千字内存，通过纸带编程。

Wolfram 告诉我，一本统计物理学书的封面引发了他的遐想，上面画着随机分布的气体分子，他编写的第一个程序就是尝试生成这幅画面。“我试着在这台计算机上解决物理问题，但没明显进展。16 岁时，我发表了一系列物理论文，离开高中，在英国政府实验室工作。”那时的“纯”理论物理研究还几乎用不到计算机，不过 Wolfram 认为，做一流的研究一定要用最好的工具——他使用一台带绘图仪的 HP 桌面计算器，并在另一台 IBM 大型机上使用 Fortran 编程。

当时的计算机主要用于数值运算，但 Wolfram 想研究的物理领域，还涉及大量代数学内容。例如从费曼图推导出表达式，成百上千个条件，都不能有丝毫差错。Wolfram 感觉自己要穷尽一生来追逐这些“负号”和“因数 2”不得解脱。他希望求助于计算机，在那时，有三套系统能完成类似工作——Reduce（使用 Lisp 编写）、Ashmedai（使用 Fortran 编写），以及 Schoonschip（使用 CDC 6000 汇编语言编写）。但这几套系统，除了它们的作者，几乎从没有人严肃地使用过，就算使用也颇为烦琐——提交一大摞卡片作为输入，经过一段时间才有结果，而得到的往往还是错误报告。

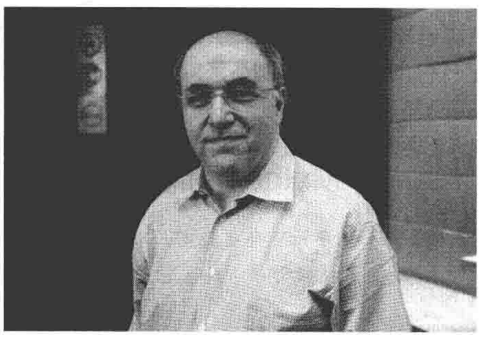
1977 年，Wolfram 在 ARPANET 上发现（那时整个网络上还只有 256 台主机），@O 236 节点是台 MIT 开放计算机，其上运行着一个名为 Macsyma 的程序能处理代数运算。此后，Wolfram 的大多数时间都花在了与 Macsyma 打交道，尝试各种可能的输入和使用方式上。

“各种奇妙的方程式开始出现我的物理论文中，旁人没注意到我使用了计算机，他们还以为我是台活代数计算器。”

Wolfram 的雄心越来越大，他成了 Macsyma 这套系统上最活跃的用户，直至 1979 年的某一天，他触及了这套系统能力的边际，他要超越 Macsyma。

那年 11 月，20 岁的 Wolfram 在 CERN（欧洲原子研究中心）思考他物理学生涯的未来。结论之一，就是若想做出更出色的物理研究，需要比 Macsyma 更强大的工具，而获得它的唯一方法，就是由自己实现——他将新工具称为 SMP（Symbolic Manipulation Program）。

Wolfram 说，他记得自己从未正式学过计算机科学，于是去书店，买下了当时能找到的每一本有关计算机科学的著作——整整半个货架，然后读完了每一本。之后，他返回加州理工学院，从世界各地邀请所有能联系上的相关领域专家，来学校演讲，他还在学校招募有相同兴趣的人，一起实现这套系统。



用何种语言编写 SMP 是设计之初的重要决定。Macsyma 用 Lisp 写成，很多人也建议 SMP 应当采用 Lisp。不过当时另一位年轻的物理学毕业生 Rob Pike（如今他更广为人知的身份是 Go 语言的设计者之一），说服 Wolfram “C 才是未来语言”，因此 1980 年，SMP 的第一行源代码是用 C 语言写下的。Wolfram 设计了整套系统，撰写了绝大多数文档，并以“每天 1 千行”的速度编写实现代码。

“SMP 在当时算得上庞然大物（虽然第一版可执行文件尚不足 1MB）。但我一点儿都没畏缩。只想着继续向前实现它。希望确保自己所做的一切都尽可能圆满。” Wolfram 回忆道。

关于 SMP 语法，那时的 Wolfram 已谙熟 ALGOL 类，以及 Lisp、APL 等计算机语言，最初的 SMP 语法明显带有它们的余风，不过渐渐地，随着他对这套系统的理解变得深刻，开始融入自己的创造，在语言设计上也变得激进。他意识到需要创造一种通用 Symbolic 语言——与组成物理世界的基本粒子相似，他试图找到用于计算的基本构件。在这个过程中，他发展出一套美学观点：将最强大的能力，封装于最少量的基本元素之中。1981 年 6 月 V1 版本发布，SMP 程序已与我们今日所见的

Mathematica 代码颇有几分相似。

彼时的 Wolfram 对商业和公司并没有清晰概念，但他明白雇用旁人开发SMP需要资金，最显而易见的方式就是出售SMP副本。他先找到加州理工学院“科技成果转化办公室”，当年这个部门只有一位和善的老人，Wolfram 询问自己作为学校的员工成立公司售卖 SMP 是否合规，在翻看了学校章程后，老人对 Wolfram 说：“学校不会宣称对软件的所有权，所以，是的，你可以。”

不过事情的发展并不顺利，学校行政部门表示“不可以”。尽管 Feynman 与 Gell-Mann 都出面帮忙调停，双方还是没能达成一致。校方表示，假如 Wolfram 仍在加州理工学院工作，同时拥有公司就不合规矩。而 Wolfram 的答案是“这好解决：我退出学校”。不久后，他加入普林斯顿高等研究院，那里的负责人告诉他，John von Neumann 去世后，他们就“把计算机捐献出来了”，所以不用担心知识产权问题。

经历了诸多波折，Wolfram 终于如愿成立了 Computer Mathematics Corporation，但他还把自己看成研究员，对如何经营公司并没有信心。他聘请了一位两倍于自己年龄的 CEO，接下来发生的事情就是 CEO 和风投，为公司安排与另一家初创公司合并，从事当时热火朝天的人工智能研发。

Wolfram 回忆说，一系列关于公司决策的噩梦就此开始，类似下面的对话经常发生：

CEO：我们开发一款运行 SMP 的工作站吧。

Wolfram：不，这是家软件公司，我不觉得咱们能开发出比 SUN 更好的工作站。

这一切令 Wolfram 沮丧，“SMP 仍是公司的摇钱树，尽管 CEO 不擅经营，他只关心拔高公司，让公司经过眼花缭乱的多轮投资——直至许多年后平淡无奇的 IPO 为止”。

Mathematica

关于愿景，假如你单枪匹马也能游刃有余，就该独自追寻。而假如你独木难支，就该做管理。愿景越大，就该做越大的管理。

——Richard Hamming

1983 年，Wolfram 退出 SMP 开发，将研究重点转向了基础科学、软件项目，以及科技和战略咨询。他偶尔还使用 SMP，但大多数时间都是用 C 语言写程序。

对于他所关注的基础科学，他设想建立一家研究中心，能让志同道合的研究者共同参与。Illinois 大学在投标中获胜，于是 Wolfram 在 1986 年来到这里建立了 Center for Complex Systems Research。

但此时，他隐约觉得自己之前“让其他人做科研”的计划不够好。于是仅在移居 Illinois 几周后便转向了“B 计划”——开发最好的工具，打造最佳个人研究环境，然后一个人尽可能多做研究。

由于之前对计算机产业有所涉猎，他预感应用软件不久之后将运行在亿万个人电脑上，若能开发出强大的科研工具，将会有不错的市场，对于他所钟情的研究，将提供莫大支持。

1986 年 8 月末，他决心开发一套“终极计算系统”，这就是今天许多人都听过的名字——Mathematica。得名于甫出走苹果的乔布斯的建议（乔布斯关于命名的理论是“首先选择一个通用术语，然后赋予浪漫的色彩”）。谈到对 Mathematica 的期许时，Wolfram 曾这样说：“如同伽利略 400 年前用望远镜观察宇宙，我希望用 Mathematica 观察可计算的宇宙”。

“如果没有 SMP，我肯定会犯许多错。SMP 告诉我哪里是切要，哪些无关痛痒，关键在于何处。”Wolfram 说，没有 SMP 的羁绊，能从头设计和实现一套计算系统，对他来说是种释放。在 SMP 中，代数计算是核心目标，而 Mathematica 则包含更多领域——数值、图形、编程、界面，还有他在复杂系统科学中的研究对象。

与开发 SMP 的过程类似，为开发 Mathematica，Wolfram 也集合了一支团队，成立了一家公司，他仍负责设计系统，仍照常编写大量实现代码，区别在于，这次，他是 CEO。

1986 年 10 月他写下了第一行 Mathematica 源代码，1988 年 6 月 23 日正式发布（这一天是 Alan Turing 的生日，但并非预先规划），Mathematica 预装到了每一台 NeXT 计算机，后来是其他工作站。Tim Berners-Lee 设计万维网使用的那批 NeXT 计算机，就是 CERN 为了运行 Mathematica 而引进到日内瓦的。

后来的事情几乎一直沿着 Wolfram 的“B 计划”发展，Mathemtica 发布 25 年后的今天，这套软件系统仍是许多科学家的首选，也是 Wolfram 和他公司的支柱，当年第一版中的不少代码，仍然原封不动地运行在最新版本中。而计划中的“基础科学研究”则是另一段故事——1981 年，Wolfram 的研究兴趣从理论物理转向了探索自然界的计算原语，他的入手点是“元胞自动机动力学”。

SMP 后世：Mathematica 发布后，SMP 仍继续销售了一段时间。Wolfram 说，所有 SMP 源代码都没用在 Mathematica 中。有一次，他想起自己手中还有一份 SMP 源代码，琢磨着何不试试在现代系统上重新编译一次？不过他随即回忆起来，自己曾经出过一个“绝佳”的主意——源代码应当加密后保存。密钥是什么？他询问了所有可能知道这件事儿的人，但没人能记起来。

一种新的科学

“还原论是对这个世界最自然的理解方式。它说的是，如果你理解了整体的各个部分，以及把这些部分整合起来的机制，就能理解整体。只要是精神正常的人，就不会反对还原论。”

——侯世达《哥德尔、埃舍尔、巴赫：集异璧之大成》

数学家拉普拉斯 1814 年断言，根据物理学定律，只要知道宇宙中所有粒子的当前位置和速度，原则上就有可能预测任何时刻的情况。

然而还原论在许多现象面前都停步不前：预测天气和气候，生物的行为、智能和思维的本质似乎都无法通过现有的物理和数学理论解释。20 世纪中叶，许多科学家意识到，这类现象无法被归入单个学科，需要在新的科学基础之上从交叉学科的角度理解。一些人开始尝试建立新的基础，包括控制论、协同学、系统科学以及复杂系统科学。

许多计算机科学家，例如 Alan Turing 和 John von Neumann 都曾为生命的奇妙现象着迷。

图灵机将“明确程序”（也就是计算）的概念进行了形式化，将计算抽象为图灵机根据机器的规则集将纸带上的初始输入转换成停机时纸带上的输出。

“元胞自动机”是理想化的复杂系统，可以简单将其理解

为一组由灯泡组成的阵列，每个灯泡的开关由其周围灯泡开关的状态决定；或者也可以理解成元素的数值根据一个局部规则在离散步中更新的数组。

Neumann 曾设计了一种有 29 种状态的元胞自动机规则，能完美复制任意元胞自动机的初始形态。他还证明了这种元胞自动机等价于通用图灵机——元胞的更新规则扮演了图灵机读写头规则的角色，而元胞阵列的状态则相当于纸带，也就是编码通用图灵机运行的程序和数据。元胞一步步更新，相当于图灵机一步步迭代。

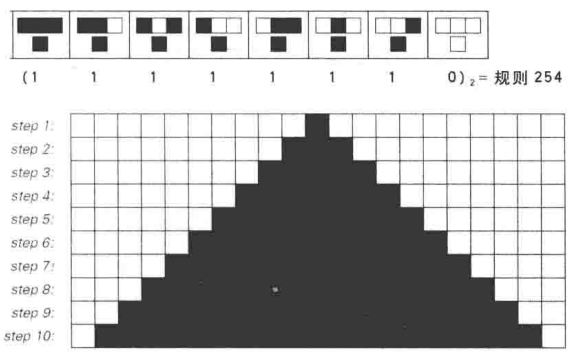
因为需要开发和改进 Mathematica，以及管理公司，Wolfram 对这方面的研究兴趣一度被压制。1991 年 Mathematica 第二版发布之后，Wolfram 终于能抽出身，他决意埋剑深山，继续先前的研究。

Wolfram 从元胞自动机的最简单形式来研究其行为，他用的是一维两状态的元胞自动机，每个元胞仅与两个相邻元胞相连，他称之为“初等元胞自动机”。Wolfram 认为，如果这种看上去极为简单的系统也无法为我们所理解，就更不可能理解复杂的多维或多状态元胞自动机了。

3 个初等元胞有 8 种可能的状态组合，每种状态又有两种可能的元胞更新状态，因此初等元胞自动机总共有 256（2^8）种可能的规则，他根据开关状态为这些规则逐一编号（编号就是对应的二进制序列）。20 世纪 90 年代，计算机的性能足以让 Wolfram 设定各种初始状态，逐个对这些规则进行研究，他用 Mathematica 绘制元胞时空图，观察变化。

他令元胞自动机运行很长时间，直至其变化模式稳定。他发现这些元胞自动机最后都进入了 4 类状态。

1. 不论初始状态如何，最后几乎都停止在不变的最终图样（例如规则 8）。



规则 254 的时空图

2. 不论初始状态如何，最后要么停止在不变的图样，要么在几个图样之间循环（最终图样依赖于初始状态）。

3. 大部分初始状态会产生看似随机的行为，虽然也会出现三角形等规则结构。其中规则 30 令 Wolfram 着迷“我意识到这幅图包含了所有科学谜团的一个线索——自然界的复杂性从何而来。”他用其来构造伪随机数发生器，并申请了专利。Wolfram 钟情于此，以至于他的名片背面，就是这一规则。

4. 有序与随机的混合：局部结构相当简单，但会移动，并以非常复杂的方式相互作用。Wolfram 认为这是最有趣的类型（规则 110 属于这一类）。

Wolfram 猜想，由于图像和相互作用的复杂性，类型 4 的所有规则都能进行通用计算。不过很难证明某个具体的元胞自动机、图灵机或其他机器是通用的。Turing 证明的只是存在通用图灵机，Neumann 也只是证明自复制自动机同时也是通用计算机。20 世纪 90 年代，Wolfram 的一位研究助手 Matthew Cook 最终证明规则 110 的确是通用的，这也许是目前已知最简单的通用计算机。

Wolfram 将规则 110 的通用性视为“新的自然定律”，并提出计算的等价性原理：

1. 思考自然界中过程的正确方法是将它们视为“计算”。自然系统正是以类似元胞自动机的方式运作，它们包含信息，并根据简单规则处理这些信息。Wolfram 探讨了量子力学、进化和发育生物学、经济学等领域，他想说明这些领域都能描述为使用简单规则进行计算。本质上，他认为宇宙和其中的万物都能用这种简单的程序进行解释——这是“广义”的计算。

2. 连规则 110 这种极为简单的规则都能进行通用计算，表明通用计算能力在自然界中广泛存在。

3. 通用计算是自然界中计算复杂性的上限。也就是说，自然系统或过程不可能产生出“不可计算”的行为。

4. 自然界中各种过程实现的计算，在复杂程度上几乎等价。之前的理论科学，总是将通过某种捷径（而不是等待完成所有步骤），就能预测系统未来的行为作为其成功的依据之一（例如可以推算出行星的轨道预测它将来的位置，而非一步步追随行星的轨迹）。而根据 Wolfram 的这一原则，对于自然界中大多数系统，复杂度不分上下，计算过程不可能找到捷径，他将之称为“计算不可约性（Computational Irreducibility）”——即使你知道系统的一切初始条件和所遵循的一切规则，计算速度也不可能比这一系统自己来得快。Wolfram 用其推论解释了“自由意志（Free Will）”。

Wolfram 走得更远，他认为存在一个简单的类似元胞自动机的规则可以作为“宇宙的终极确定模型”，它是万事万物的源头。这个规则有多长？Wolfram 说：“我猜其实相当短，也许就三四行 Mathematica 程序。”

这些内容都写进了 2002 年出版的一本将近 1200 页、5.6 磅的大部头——《A New Kind of Science》（NKS）。

“写作这本书，我超过十年几乎连续不断地工作，敲击了过亿次键盘，移动了一百多英里鼠标，积攒了数十 GB、万页笔记，运行了近百万行的 Mathematica 程序，执行了兆亿次运算。”Wolfram 在序言中这样写道（从 1989 年，Wolfram 就开始记录所有邮件、作息、敲键、电话、运动等个人信息，他认为这些信息不但为我们感受生活增加了一个新维度，还能用于分析过去，预测未来）。

作品完成后，Wolfram 曾邀请乔布斯在封底写一段推荐语，乔布斯说“牛顿著作的封底没有推荐语，为什么你需要呢？”所以 NKS 最终以简洁的图片作为封底（Walter Isaacson 的乔布斯传记《Steve Jobs：A Biography》也是相同的风格）。

“三个世纪前，人们用数学描述自然界的思想，使科学发生了极大转变。在这本书中，我将尝试发起另一次转变，建立一种新科学，它基于更基本的法则，并且可以通过简单的计算机程序将其具体化”——Wolfram 在开篇中这样写道。

“科学上的进展都是一小步一小步地演进，如果不依赖已有的特定领域知识，很难解释清楚。但为了在书中描述一种新的科学，我不得不选择跨越几大步，因此，我需要从零讲起——

那些很少依赖过去的新思想和新方法。”与其他划时代科学巨著不同，阅读这本书并不需要太多知识背景，整本书也很少见到数学公式，几乎都用图形进行推理和证明，这是一本大众也能读懂的科学书。

书中首先用元胞自动机完成了多种数学运算模拟，并将一维元胞自动机扩展到多维，模拟更复杂的雪花、生物细胞，再进一步将它与周围的真实世界联系起来，例如交通堵塞、股市涨落。尽管外界对 NKS 毁誉参半，但并不影响这本书的流行。

《A New Kind of Science》出版十年后，Wolfram 着手为它增加新的一章，在尚未出版的手稿中他这样写道：

新科学的阶段：

- 吸收（尽量理解书中所写，初步理解需要两年，进一步需要 5 年）；
- 初步延伸（2 ~ 3 年，10 ~ 15 年后结束）；
- 建立主要新方向（15 ~ 30 年）；
- 小规模早期技术应用（4 ~ 10 年）；
- 大规模技术应用（10 ~ 25 年）；
- 成为常识（4 ~ 10 年）；
- 成为基础科学教育的一部分（15 ~ 20 年）。

原则：

- 寻找最简洁的案例，解决最显而易见的问题；
- 不要满足于技术解释，探寻事件的根本原因；
- 不要被前人的工作束缚，但要尽可能全面理解这些工作；
- 尽可能清晰地解释你的工作，少依赖底层架构。

这或许可以作为 Wolfram 科学生涯的注解。

生物与计算机交织的独特人生

——Facebook HipHop 作者、阿里研究员赵海平专访

记者 / 张勇

3 月 26 日，杭州的天阴沉沉，这是一种山色空蒙雨亦奇的美丽，还是雾霭笼罩下的怪异，对于来访阿里巴巴西溪园区的人们来说，没人关心这些。人们行色匆匆，兴奋地往各自目的地奔赴而去。我也来不及细思这些，因为今天要和刚从 Facebook 来到阿里的赵海平聊天。

赵海平是一位知名的软件工程师，曾在微软工作过。2007 年加入不到 50 位软件工程师的 Facebook（是第一位中国工程师），期间他创建了 HipHop 项目。HipHop 可以将 PHP 脚本代码先转换成抽象语法树（AST），之后再转换成优化的 C++ 代码，使其速度提高 5~6 倍，为 Facebook 节省了数十亿美元。2015 年 3 月他回到中国，加入阿里巴巴技术保障部，重点攻克阿里在软件性能以及 Java 使用过程中遇到的技术问题。

采访在园区图书馆进行，四周书籍环绕长窗落地，赵海平看上去显然刚从另外一场繁忙的事务中抽身过来，但在这场长达 1 小时 47 分钟的采访中，他一直神采奕奕、兴致高昂地谈论了各种话题：小时候的趣事、生物和计算机间的痛苦抉择、HipHop 项目中的艰辛……

“计算器有什么好学的？”

赵海平中学时代就读于秦皇岛市山海关第一中学，学校虽然非常小，但很特别——恰好在天下第一关脚下，所以长城就成了这个学校的一面校墙，坐在部分教室里甚至能领略到山海关的雄姿。

在他的那个时代，计算机还是个稀罕物，别说是高中，可能在大学，计算机都很少见。所以当他们北大物理系毕业的校长组织数学好的同学，参加学习计算机的课外活动时，赵海平很是疑惑，“计算器？这个需要学吗？”事后赵海平才知道，他把“计算机”听成“计算器”。不过那个时候他真不知道什么是计算机，并且《计算机报》也是在几年后才出来，就连当初学习计算机的时候，整个书店也只有一两本计算机相关的书籍，而内容早被他们翻烂。

起初学习的过程很原始，“一开始连计算机都没有，学校虽然已经去买，但要等很长时间。所以学 Basic 语言时，完全不知道在干嘛，就纯粹硬学，学到最后连循环都学了，还没见到计算机。”后来计算机到了——是 Laser-310，赵海平对这个记得特别清楚，谈到这里的时候，他还绘声绘色地形容“一按那个键盘，还‘哔啵、哔啵’响”。赵海平回忆称，当时的游戏也很简单，简单到只有小人在屏幕上又唱又跳。放到现在看这哪是游戏，但在那个时候觉得很奇特。

小时候学计算机发生了两件事让赵海平记忆犹新，一是利用汇编命令打印系统。大家都知道利用汇编命令可以把删掉的文件再找回来，那个时候的赵海平觉得这很牛，于是去找汇编命令然后到学校实践，却把整个系统都给打印了。他至今仍很兴奋地说道：“苹果有反汇编的工具，它可以不断地反汇编操作系统，所以一边反汇编一边打印。机房老师不知道这事，但机器却一直在打印，打印了一宿，把机房的一摞纸全用了，而那个时候的打印纸特别贵……”

另一件事则是，“废寝忘食”地输入飞机。当时整个学校只有一台计算机，赵海平每天中午都是赶紧扒完饭，省出一两个小时的时间去机房。有次在机房按照杂志上的坐标输入显示飞机。“图的打印很简单，实际上就是从这一点到另一点画一条线，但它有很多条线，最后能画出一个特别漂亮的飞机图。”没想到快要结束时，有人碰掉了把电线，内容全没了，大家面面相觑，又心有不甘。于是第二天又跑到机房重新输入一遍，最后看着苹果绿颜色的屏幕上呈现的飞机，赵海平觉得那种成功的感觉至今仍很兴奋和微妙。

抉择：继续生物 or 计算机？痛苦！

如此喜爱计算机，为什么最后去了北大生物专业？这应该是很多人的疑惑。赵海平称，没有选择最爱的计算机专业

有两个原因：一个是生物在当时太热门——21 世纪是生物的世纪，而当时赵海平高考的分数恰恰很高，“要低一点就能报计算机系，但高一点的全都直接进生物系，所以当时各个地方的高分学生都去学生物了。”第二个原因则是大家对计算机的认识。“那个时候还认为计算机只是一个工具，不管做什么，你都会‘玩到’计算机。”大家都把计算机当玩具，不是一个正儿八经的专业。

不过当学了生物很长时间之后，赵海平发现自己还是喜欢计算机，他这么描述当时自己的感受：“我不喜欢生物，也做不好生物。身边学生物的人可能不比我更聪明，但他们做得比我好，因为他们热爱这个专业，愿意花很多时间去看文献做实验。我更多的是在生物实验室里写程序，找与计算机相关的活……这是一种很别扭的感觉。”于是他决定踏上痛苦的转行之路。

之所以痛苦，主要是不愿意就这么放弃生物，用赵海平的话来讲就是“生物系的机会也很宝贵，而且已经学了这么久，又不确定自己的将来做计算机会不会更有意思？”另一方面则是生物实验室的台湾导师对赵海平格外青睐，他知道赵海平很聪明，但心思并没放在生物上，不过他仍然希望赵海平能够继续把博士读完……

这些因素让赵海平一直很痛苦地思考这个问题，直到一件事让他下定决心转专业。那时候他已在美国纽约大学，“一上选修的计算机课感觉喜欢得不得了，听老师讲东西都有一种触动——原来是这样。”后来选修课的老师也鼓励他去学计算机，于是他下定决心，并整理自己做过的小项目去申请普林斯顿大学计算机硕士。

半听半睡状态下，突然举手“老师这不对”

进入普林斯顿计算机系后，赵海平如鱼得水，门门功课都是 A+，学习过程更像是在印证他以前的实践，并补上自学摸索中所缺失的东西。“来到普林斯顿后感觉特别好，不仅仅是因为终于学了自己喜爱的专业，还有机会和非常优秀的人在一起，觉得特别快乐。”

谈起在普林斯顿最大的感触，赵海平称，好几个同学都是当年奥林匹克金牌得主，虽然年龄较小，但都非常聪明和优秀。“所以我们工作一定要进入一家好公司，因为跟优秀的人在一起工作，就自动地有那种很快乐的感觉。”

在普林斯顿学计算机的赵海平同时也很疯狂：经常打游戏、做小项目，有次到第二天早上五六点才睡，但发现八点还有课，于是他在那半听半睡，听到半途突然醒了，举个手说老师这不对……在这样的精神状态下，他还不忘给老师挑错，而老师也觉得，这学生挺神奇，一边睡还一边提问题。

原本赵海平打算读完计算机硕士，然后再把生物学博士读完，但在毕业前的最后一个星期，他就拿到了四个 Job Offer。是读博，还是工作？赵海平又陷入了抉择，但最后还是听取了生物实验室同学先工作的建议。

“现在回想起来，这是在骗自己。”

为什么？

“一旦干上计算机这行，生物博士对我并不是特别的重要。”赵海平如此坦言。

赵海平指出，如果有些活是重复性的，一定要思考怎么才能偷点懒，花点时间去想，你就能比别人快。

不后悔读生物，非常感谢生物的训练教他如何系统“偷金子”

对于赵海平改行学计算机，不少人都有误会，因为那个时候计算机恰好也火了，很多人以为他是因为计算机火了才改行。“其实真的不是这样子，那个时候 MBA 更火，工资也更高。也有朋友劝我去读 MBA，但经验告诉我一定要做自己喜欢的事，不能再跑到第二个不喜欢的行业，不然这辈子就废了。”而这也是赵海平改行之后，为什么老是如饥似渴地去学习和工作的原因，因为他总觉得耽误的时间特别长，心理上想去补偿。

但对于读了这些年生物，赵海平并不后悔，他称读生物的那些训练非常有用，对他帮助很大。“它教会了我，怎样系统地去解决一个问题，而不只是把这个问题解决。”赵海平非常善于举例子，他打了个比方：如果我们是一群海盗，有一个岛有很多金子。本科毕业的学生划着船弄来一箱金子就很出色；但如果是博士毕业的学生去弄金子，除了弄来金子之外，他还必须带来一句话才合格——没必要再去那个岛了，金子全被拉回来了。“这就是博士的训练，它告诉你要系统化地去解决一个问题。等解决问题后，你已经看到问题的所有方面，等别人再想解决，你已经全部想到。”

赵海平放弃生物学博士学位，虽然有些许遗憾，但他庆幸的是这些训练没有白费。

微软如日中天时去了 Facebook

2006 年前后，微软如日中天、风头正劲，而当时的 Facebook 全部工程师还不到 50 人，在这样的情况下，赵海平离开微软，去了前景仍不太明朗的 Facebook。

离开微软，主要是两个原因。一个是微软当时的流程非常规范，强调人和人之间按部就班做事，非常条理化。这也许是大公司发展到一定阶段的必由之路，但赵海平希望寻找节奏更快的平台，“毕竟我在生物上已经耽误 10 年了。”赵海平解释道，“当时的微软和硅谷公司在文化上还是有明显差异的，硅谷文化强调个人的主动发展远远大过公司给你规划好的发展。”另外一个原因则是家庭因素，最终他来到了 Facebook。

Facebook 的经历太独特、珍贵

那个时候 Facebook 人特别少，只有 40 多个程序员，大家都叫工程师，没有分组，所有人都挤在一个特别拥挤的办公室。桌子很小，彼此离得很近，出了问题，一吆喝就行。Facebook 用户数增长得特别快，大家的工作也都围绕着快速增长带来的问题进行，“一旦有什么问题，大家就往上扑，你愿意解决什么就解决什么，特别没有章法。”赵海平称，这恰恰和微软形成一个鲜明的对比——毫无章法，乱到你有时候觉得微软还不错，挺有章法。

赵海平在 Facebook 待了 8 年多，他称无论是用户的增长、公司上市的奇迹，还是 Facebook 对整个世界的影响，都太独特。当初没有人想到，至少赵海平以及他身边的同事都没想到 Facebook 会是今天这么大的规模——当时只是看到在不断增长，并不知道增长之后意味着什么。独特也还包括做项目的艰辛，“我们是被逼地把网络和服务器做到极致，因为每省 0.1%，就是几百万美元，所以必须把东西做到最好。”

现在回头看这些经历真太特殊，当年坐你旁边的人都成了高管，这是一种特别奇妙的感受，“像当年 Facebook 的联合创始之一 Dustin Moskovitz 就坐在我旁边写过程序，你现在让他再坐在我旁边写程序，不太可能！”但当时大家没有想这么多，所有人都干得热火朝天，讨论问题可能会一直持续到十一二点，下班了也可能在工作——网页怎么设计、后端应该怎么做……一大堆意见。

这么疯狂的原因很简单，一个是 Facebook 用户不断地增长，一直在鼓舞着大家；另一个则是这些人都非常优秀——哈佛、CMU、MIT、斯坦福……每一个人都特别有才、特别有想法。

到阿里是想拥有另一段独特的经历

从 Facebook 到阿里，并不是一个很突然的决定。赵海平加入阿里之前，已经跟阿里技术保障刘振飞、周明、毕玄聊过，对阿里已经有了整体的了解。

正式接触是在 2014 年阿里 IPO 前后，阿里组团来美国，王坚博士和赵海平深入聊了聊。发现技术很对口，需要解决的问题也很新鲜——Java、JVM 从来没做过。他又来了趟杭州，发现杭州挺美、阿里员工脚踏实地的公司文化很符合他的性格，对阿里很有好感。

而让他下定决心回国则是阿里面临的技术难题和挑战。“在 Facebook 经历非常特殊，而阿里就更特殊——大规模的交易、单天的交易量、双十一的交易量，不是任何一家美国公司有的，这种独特性是我寻找的东西。当你去解决这样的技术难题时，就会有新的认识：这个量级的问题可以这样解决，提升一个量级，就要用新的解决方法，最后你就能看到问题的全部。”体会从不同角度去看不同的技术难题，对赵海平来说恰恰就是最享受的一件事情。

HipHop 不为人知的故事：一开始野心并没有那么大

其实最初做 HipHop 项目时，他们的野心并没有那么大，不是一定非要把它转换成 C++，但后来发现，这么做确实可以提高性能，所以就思考是否可以用工具把整个网站都编译。

那时 Facebook 内部有好几个项目都处于半竞争关系，每个人都仁者见仁、智者见智，有人想改变 PHP 引擎实现，有人想在 PHP 应用层做调整，有人说可以转成 Java，而赵海平则倾向于保持 PHP 不变。“之所以倾向于不变，是之前大家已经尝试过将 PHP 转换成其他语言（Python），但四五个人转，根本赶不上二三十个人写 Code 的速度。”另外，团队里有很多 PHP 专家，突然改成 Java，让人怎么写代码？于是赵海平一直在这个领域探索，到最后只剩下 HipHop 和转 Java 两种方案。这时大家意见又不一，到底用哪个？讨论问题也有了情绪，怎么办？用数字说话！谁性能提高大，就用谁，最后 HipHop 胜出。

线上执行的时候，赵海平内心很忐忑，很怕出现什么问题。“因为你改的是底层，只要有一个新问题出来，大家第一反应就是这里出了 Bug，事实上十有八九都是应用本身的问题。”当时赵海平对其他团队的承诺是，有任何问题 24 小时内一定响应。有一年感恩节，正在别人家做客的赵海平，一个电话就立刻赶回去调试，虽然事后证明也不是底层的问题。“当时我跟团队说，我们要有阿信卖鱼的精神，不论是哪里的问题，我们都要帮他们 Debug，赢得别人的信赖。”对于这些，赵海平虽然感觉非常辛苦，但又感觉很爽，因为当时他们可能是最熟悉全部应用的团队。

再回首 HipHop，赵海平感慨项目非常艰辛。“最累的时候，我开车都在想怎么写 Code，一边开车一边想，到了公司只是把脑子里想好的写下来而已。那个时候别人跟我说话，我都不大反应的过来，因为脑子一直在想这个东西。没有办法，面铺得太大了，等于整个程序语言的所有都要去实现。有时也想偷点懒，想着这个数据库函数这么冷门，肯定没人调用，先不做它，结果第二天就有人来找你……”

虽然过程很痛苦，但痛并快乐着。赵海平找 Bug 每次都很快，半小时、一小时就能找到。但有一次 Bug 太底层，花了十几个小时才找到，最后找到 Bug 跟大家解释怎么回事时，那快乐真的很难用语言形容。”赵海平接着补充到，“其实人有的时候就是喜欢解决大问题、大影响的那种快感，希望自己做的工作特别有意义——掷地有声的那种。”

性能优化要先剖析、迎着问题走、用数据说话

谈到即将在阿里的工作，赵海平称，他不会急于跳进很具体的东西里去优化。他首先会从整体上做剖析，分析阿里各方面软件的运行情况——究竟用了多少资源。看清楚问题，才能钻进去优化，也只有这样才会事半功倍，才能真的找到最大优化的那个点。

不做剖析就去优化是一个误区，你花了特别大的力气优化，速度提高一百、一千倍，但整体上可能还是没有效果，因为这一块性能只占整体的 0.1%，你提高一万倍、甚至把资源占用降为零，最后你也只能提高 0.1%。

优化一定要用数据、效果说话，有个误区是有人会认为某种语言就是比其他语言强，所以优化上总是选择某语言。但

在实际场景中不能这样，比如：不是说 C++ 一定会比 Java 执行得快，在有的情况下 C++ 的速度可能和 Java 一样，甚至比它慢。所以在优化的时候，一定要用数据说话，不能凭着自己的技术信仰做决策。优化的方式可能仁者见仁、智者见智，但最关键的是大家认同这个问题，然后用各种方法试，谁的效果好，就用谁。

在性能优化上，一定要迎着问题走。不少人看到问题后，会想办法绕开问题——不调用或用其他做法，但专家会迎着走，想办法解决问题。比如说网络延迟很高，专家会钻进去寻找延迟的原因，网卡有问题就解决网卡，软件有问题就改软件。赵海平总结道："越迎着问题，问题可能解决得就越好。"

优化上的取舍难、行政难

在性能优化上除了一些误区之外，还有一些其他困难。其中一个就是取舍的问题：知道怎么优化，但有代价，比如提升了一个地方的性能，可能影响了另一处的性能；或者今年和明年的情况完全不一样，今年是利大于弊，但第二年可能就弊大于利，这是取舍难问题，要想解决就必须实时跟踪系统不断地做调整。

赵海平看来，最难的就是整体性慢。"整体性慢就是哪都不慢那么多，但哪都慢，这才是最要命的一种情况。"如果遇到这种情况，就要仔细想想更底层的东西，究竟是什么原因让它普及性得慢？这个时候也许你能发现更大的问题。

赵海平比喻称，这就像有些病人的病根很深，所以表现出各种各样的症状，但各种症状的根可能是在一个地方。"这就是我想说的：要想系统地提高性能，有时候要做架构上的大调整，进行深入的过程性优化，而不是一个具体、很有局限性的优化。"不过走到这一步，就会出现新的挑战，"不仅仅是技术上的难，还可能会出现行政上的难——你如何去说服别人把整个系统做大的调整。"赵海平感慨道，"这可能会影响到线上的服务质量，换那么多新东西，大家也不知道能省多少，这个有时很难做。"

很多技术专家擅长解决技术上的难题，那遇到行政上的难如何解决？赵海平也给出了他的看法。"项目刚开始野心做得不要太大，从小的地方做起，让人家看到一些实实在在的好处，然后你就能做下去。而能滚多大，就要看它是不是每一步都能带来很大利益，如果滚不大，就别做了。因为你确实没有办法去说服别人，不能说眼下没有利益，我现在花钱，过一年省钱…这个很难实现。"

对语言之争的看法："有人会去争论锤子和斧子哪个更好使吗？"

聊到程序员对语言信仰的问题，赵海平称，不光国内，在美国也一模一样，这个可能就是程序员的特点，喜欢和人争论哪个语言更好用。从某种意义上来说语言也是一个流派，喜欢同一语言的人，往往会有比较明显的共同点。

但他很疑惑语言之争："有人会去争论锤子和斧子哪个更好使吗？它不是要根据你做的事来定吗？"赵海平认为："不同的语言像锤子和斧子一样，没有哪个可以把所有的问题都解决好，每一种语言都是在解决某一个特定的问题里有它的长处。这个恰恰是语言创造者的思路，他有自己认定的语言要解决对应问题的范畴。你偏要去争论某语言就是绝对的通杀，比其他语言样样都好，这个肯定不对。当然你可以有个人的喜好，但不能非要改变别人的想法，因为语言本来就是百花齐放、百家争鸣。"他还指出，语言之间的差异，只有在解决某些具体问题时可以比较出来，而这些恰恰就是语言独到的地方。

如果非要说哪个语言类似于万能语言的角色，赵海平认为目前应该还是 C 和 C++。"它很底层，不做太多假设。越是这样的语言，它越通用一些。只要语言做了假设，它就会更上层一些；只要有假设，就会放弃掉一些东西，因为假设让它更加有局限性。"像 Java、PHP、Python 等其他语言，它都提前实现了一些东西。在帮你提前实现东西时，它已经做了假设。"比如说 Java Applet 定义已经做了假设，PHP 里数据类型已经做了假设，这实际上相当于定了一个框架，只能用它那一套内存管理和执行环境。但在 C 和 C++ 里面没有这些问题，你自己可以定义这些东西。这也就是为什么越往后端的东西，就越要拿 C 和 C++ 写的原因，因为对系统要求独特，不像高层的软件用这些假设没问题，底层的东西很多时候是不能用假设。"

但缺点也在这里——你要花时间定义，可能你定义来定义去还没有人家 Java、PHP 或 Python 等其他语言好。

对开源的认识和思考

赵海平的 HipHop 给 Facebook 带来了很大影响，为了惠及他人和社区，最后把它开源了。那他对开源有哪些认识和思考？

就开源还是不开源，赵海平拿 Facebook 和 Google 举例。他称，Facebook 开源做的特别多，而 Google 做得挺少。主要原因是 Google 有很多关键、核心的技术不能开源，因为一旦开源可能会给它带来竞争压力，而 Facebook 没有这种压力，因为就算拿走它的源代码，也不意味着能够复制出一个 Facebook。所以赵海平认为，只要公司不受影响，你就可以开源。而开源带来的惊喜和好处很多，"如果很多人关注你的项目，至少说明做的方向是对的，别人拿去执行，反馈过来的问题相当于帮你做 QA。如果人家特别感兴趣，深入研究、甚至帮你做修正，这等于免费帮你打工……"赵海平还发自内心地谈到，很多开源社区的人思考问题方式很独特，和他们聊天很能扩展你的视野。

总体上来说，开源是一件利大于弊的事情，虽然它可能会耗费你很多时间和精力，但在耽误时间的背后，它有巨大价值：学术上带来名气、能提高你的业务水平（别人的指指点点就是学习和提高的过程），而从大的角度来看，也给公司创造了技术开放、乐于回馈的良好形象。

专注才能走得更远

在很多人眼中，赵海平已然是一名伟大的计算机科学家，所以在采访中记者也就他如何走到今天的高度以及生活中的一些习惯进行了交流。

赵海平很谦逊地表示："什么是伟大的科学家？伟大的科学家必须能够为科技带来革命性的变化，例如创造了新

型的语言或操作系统，我只是做了一个项目，而且只是对Facebook意义很大，所以还差很远。”他又继续分享道，“不过要想做成事，心态很重要。我来的时候就跟同事聊天说，要忘记过去的成绩，脚踏实地做事，可能你比别人有经验，但还是要扎扎实实展开工作，要把自己的注意力转移到要解决的技术难题上、转移到个人的知识扩展上。你专心地去做这些事情，不去想太多其他事情时，可能会走得更远、更好。”

他还就专业的程序员话题举例，“我不相信麦迪（Tracy McGrady）在最后投三分球时，他脑子里会想比分、输赢、会去想这场比赛能给他带来多少奖金，他不会去想这些，只要想了三分球就投不进去。他要的是那种完全投入的境界和状态，只有在那种巅峰的状态，他才能把球打得最好。”任何一个职业运动员都懂得这一点，专业的程序员也同样需要做到。只要做得好，别人就会认可，一切东西都会水到渠成。

高效率诀窍：一整段时间非常重要

一个人要想取得成就，一方面跟他的天赋有关，另一方面他的习惯也不可避免地起了部分决定性作用。在个人习惯上，赵海平非常强调专心、强调一整段时间。他写程序的时候希望至少四个小时内不被打扰，他也不去看手机、Email、接打电话、参会等。这是一个高效率的诀窍，“因为写程序是高度专注的脑力活动，你一被打扰就转出去了，再转回来不仅要花时间，而且这种转换消耗特别大，所以要尽量避免或减少被打扰。这也是一个很好的用脑习惯，只要养成这个习惯，有的时候你会发现写程序也是一种休息——当你进入状态时，你不会觉得累。”

另外，每个人也要总结一套高效率的工作方式，比如说用什么样的编辑器，在这个编辑器下敲什么样的命令可以省时间，尽量多思考一下。“这方面我比较变态，我有很多的命令都是一个字母。‘啪’，敲一个字母命令就出来。”赵海平指出，并不是非得要这样做，他只是想强调：如果有些活是重复性的，一定要思考怎么才能偷点懒，让复杂的事情三两下就做完，花点时间去想，你就能比别人快。

对在阿里的未来：会去思考怎样做这些贡献

在赵海平那封告别信中，他曾期待，他想要中国成为软件技术最好的地方，想要阿里成为最值得工作的地方。为了这个目标，他会怎么做或进行哪些努力？

赵海平称，一方面希望把技术工作做好，期待能给阿里带来整体性能的提高，能够真的建立一整套剖析系统，大家都能够根据这个系统做软件上的调整，从而提高整体性能、节省机器。

另一方面希望带动大家对技术的爱好，带动一批人、带动气氛，并且也期待有一帮好哥们。如果还能进一步的话，就做一些开源，给阿里创造开源的好形象，带动整个中国的开源氛围，提高各个公司对相关技术上的认识……

而在未来，他有可能也会做一些中美技术交流上的牵线搭桥。他还特别强调：“不希望大家把这话理解成我会做到这些，只是想和大家表明：我会去思考怎么样去尽量做这些贡献。”

结束语

采访最后，记者还就2010年PHP之父Rasmus Lercbrf对HipHop的一个评价，询问了赵海平的看法。Rasmus Lercbrf当时评论：不能把HipHop当作银弹，它确实很酷，会成为某些网站很好的选择，但对于很多Web应用而言，执行速度并不是主要因素，所以不能对任何项目都适用。

赵海平回应称，Rasmus Lercbrf说得非常正确。“他想说的意思就是：要先剖析你的软件，然后再做性能提高，他表达的和我说的是一样的。”他进一步解释道：“之前Facebook软件剖析发现PHP的CPU用得很多，所以HipHop才管用。但如果你剖析软件，发现更多的时间花在网络等待上，而没有花什么CPU，那么用HipHop怎么可能提高你的性能？”赵海平还表示，Rasmus Lercbrf的评论说在了一个敏感的时刻——正好在我们发布时，这看起来似乎有点负面，但并不是这样子。“他说的非常科学化，就是想告诉大家一定要完完全全理解你的软件，究竟是网络时间，还是CPU时间，这是截然不同的两个概念，在性能优化上也是截然不同的途径去解决。”

CTO要30%懂产品、30%懂管理、40%懂技术

——快的打车联合创始人兼技术副总裁闻诚专访

记者 / 夏梦竹

快的打车成立于2012年，发展至今已覆盖全国360个城市。近期，CTO俱乐部采访了快的打车联合创始人兼技术副总裁闻诚，请他分享“快的”的蜕变之路。

闻诚，2007年浙大硕士毕业，随后在一家外企工作5年，当时主要负责移动互联网方面的产品。用他的话来说“尽管人在外企，但心里总想着能做点什么”。那时的他就认定未来移动互联网一定是大趋势，由此有了创业的想法。

一路走来，每一步都很精彩！

CTO俱乐部：回头看这一路的努力和经历，有哪些让你最为难忘的人或事？

闻诚：创业的经历其实每一步都非常难忘。和在公司中打工不同，创业的每个困难都要去克服。因此，创业方向非常重要，所做出的每个决定都有非常大的影响。经过不同的尝试，发现一个最有可能正确的方向，做最正确的决定。这个过程对自己能力以及团队等各层面都有极大的挑战。

2014年初闹得沸沸扬扬的打车大战，可以说是一场突然爆发的战争。这场战争中，快的和对手在技术上都没有做好充分的准备，在打车大战期间，用过快的打车的同学可能会发现有服务不可用的情况发生，这就是我们所经历的一次非常大的考