

架构师

ARCHITECT



热点 | Hot

Java 10新特性前瞻

Linus怒喷谷歌安全工程师

推荐文章 | Article

阿里正式开源其自研容器技术Pouch

Kafka不只是个消息系统

为什么原生应用开发者需要关注Flutter

观点 | Opinion

杨帆 : AI落地的关键是算法闭环



CONTENTS / 目录

热点 | Hot

Java 10 新特性前瞻

阿里重启维护 Dubbo 了

Linus 怒喷谷歌安全工程师

推荐文章 | Article

阿里巴巴正式开源其自研容器技术 Pouch

Kafka 不只是个消息系统

为什么原生应用开发者需要关注 Flutter

观点 | Opinion

商汤科技杨帆：AI 落地的关键是算法闭环



架构师

2017 年 12 月刊

本期主编 覃 云

流程编辑 丁晓购

发行人 霍泰稳

提供反馈 feedback@cn.infoq.com

商务合作 sales@cn.infoq.com

内容合作 editors@cn.infoq.com

卷首语

当我们在架构设计的时候，我们到底在做什么？

作者 得到 张明庆

我们为什么要进行架构设计？如果我向架构师征集这个问题的答案，我想可以收到成百上千条。这些答案绝大多数都是对的，但是可能都是一些理性的回答，例如架构可以让代码更稳定、更高效或者更富有弹性。如果抛弃这些理性的分析，从感性上来讲，是什么驱动着我们克服重重困难去做架构设计这个苦差事呢？

我想这就要归因到我们成为程序员的初衷，我们投身到互联网这个行业，就是因为这个行业总是在追求对世界的改变，并且是向好的方向改变。不论是让人们的生活更便捷，还是让这个世界更透明更安全，这些改变都深深地吸引着我们，因为作为程序员的我们就希望生活在一个便捷透明的世界里面，或者说我们都希望拥有一个更简单更从容的人生。

再回来看我们上面列出的答案，架构设计不论是力求简洁高效，还是追求稳定和弹性，这不就是一个程序员世界观和人生观的映射吗？或者说我们在进行架构设计的时候，我们其实在按照自己的内心设计一个小的

世界，一个一定程度上由自己说了算的世界。

让我们来看看，在这个说了算的世界里面都有什么？

1. 这个世界里面有“墙”，我们希望能把代码梳理清晰，各有所属，各司所职，并且剔除坏的代码，让好的代码发挥更多的效用。解耦是架构设计永恒的主题。
2. 这个世界里面还有“路”，路把代码有机地联系在一起，信息在路上自由地流动，互通有无，我们希望沟通简洁而有效，因此每条路都清晰而有序。
3. 这个世界里面自然还有“人”，除了作为建设者的程序员，更多的是这个系统的其他参与者，对应的是整个系统各个环节上的伙伴们，有产品有运营有运维也有测试。一个完整的架构设计，需要让所有的人都可以在这个世界里面找到自己的位置。
4. 有了人，也就有了制度。正如康威法则所揭示的，组织架构和代码架构之间存在着映射关系。我们在设计的时候需要参考组织架构，但同时我们也希望一个更好的代码架构可以反过来推进组织架构以及业务流程的改变。任何架构的设计，包含了架构师对外在环境的感知，自然也蕴含了其对外在环境的期许。

也许评价一个架构是否合理，也是一个感性的事情，如果这个架构契合了内心希望的那个世界，这种架构就是程序员所推崇的。很多情况下，我们要大刀阔斧的去进行架构设计，或者孜孜不倦地调整架构，就是因为架构的内在和外在还有我们不能忍耐的地方，就像我们不能忍耐生活中的某个方面而总是寻求着各种改变一样。



全球人工智能与机器学习技术大会

助力人工智能落地

2018.01.13 – 01.14 · 北京国际会议中心

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

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

部分演讲嘉宾



颜水成
360人工智能研究院
院长及首席科学家



山世光
中科院智能信息处理
重点实验室常务副主任
中科视拓董事长/CTO



袁进辉 (笔记本)
一流科技
创始人



刘海锋
京东商城
总架构师&技术VP



洪亮勘
Etsy
数据科学主管



于磊
携程
基础大数据产品团队总监



张浩
饿了么
技术副总裁



尹大朏
摩拜单车
首席科学家

精彩案例抢先看

摩拜 | 如何使用人工智能实现单车精细化运营

知乎 | 如何使用机器学习实现News Feed正向
交互率提升100%

国美 | 推荐引擎与算法持续部署实践

微博 | 深度学习在红豆Live直播推荐系统中的应用

Tutorabc | 大数据和AI之路

饿了么 | 机器学习和运筹优化在外卖行业的应用实践

第四范式 | 如何利用大规模机器学习技术解决
问题并创造价值

微信小程序 | 商业智能技术应用实践

爱奇艺 | 自然语言处理和视频大数据分析应用

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

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



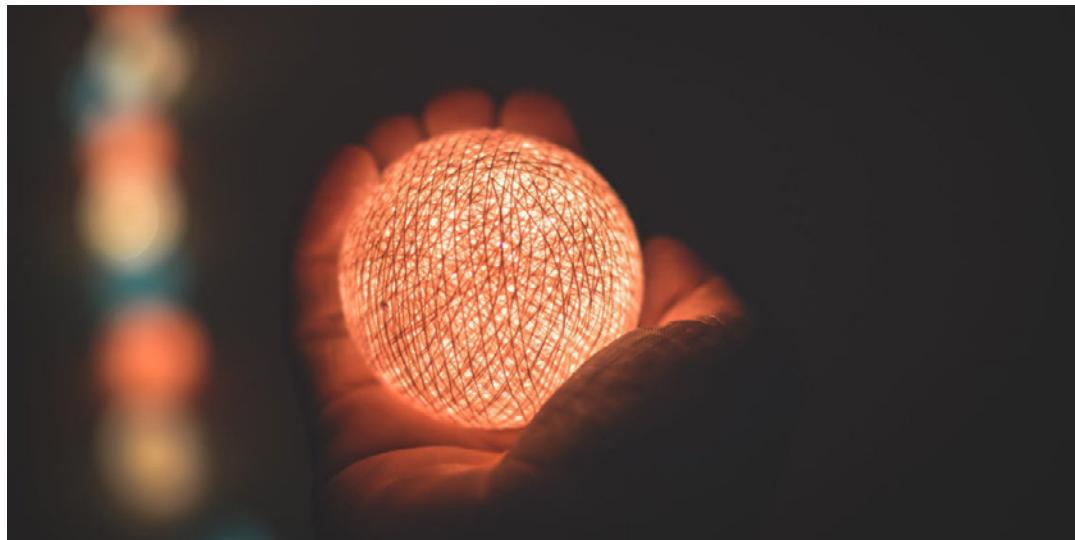
购票请联系



扫码关注大会官网
获取更多大会信息

Java 10 新特性前瞻

作者 Ben Evans 译者 薛命灯



从 Java 9 发布到现在已经过去两个月了，根据最新的[发布计划](#)，距离下一个 Java 版本发布只有四个月时间。Java 10 的新特性还在确认当中，所以从现在到 GA 版中间还是有可能加入重大的变更。不管怎样，在这四个月里，开发者还是可以期待一些新的特性能够被添加到 Java 10 中。

新的特性和增强一般通过[Java Enhancement Process](#) (JEP) 或 [Java Community Process](#) 标准请求 (JSR) 进行跟踪。因为 Java 10 的时间线较短，范围也相对较小，所以 Java 10 的变更将通过 JEP 进行跟踪。

有望被包含在 Java 10 中的特性是那些已经处于 Targeted 或 Proposed 状态的 JEP，它们包括：

- 286: 本地变量类型推断
- 296: 统一JDK仓库
- 304: 垃圾回收器接口
- 307: G1的并行Full GC
- 310: 应用程序类数据共享
- 312: ThreadLocal握手机制

JEP 296 是一次纯粹的清理工作，而 JEP 304 加强了不同垃圾回收器的代码隔离，并为垃圾回收器引入更简洁的接口。

JEP 304 意味着厂商可以更自由地选择特定的 GC 算法来构建 JDK，因为现在有多种处于开发当中的 GC，如 [Shenandoah](#)、[ZGC](#) 和 [Epsilon](#)，在未来可以使用这些 GC 算法。社区也在努力[弃用甚至移除 Concurrent Mark Sweep \(CMS\) 垃圾回收器](#)，只是目前还没有可用的替代品。

比较有意思的变更或许是 JEP 286，增强的本地变量类型推断可以让开发者免去很多变量申明模板代码。也就是说，在下一个版本中，下面的变量声明是合法的：

```
var list = new ArrayList<String>(); // infers ArrayList<String>
var stream = list.stream();           // infers Stream<String>
```

这种语法只限于初始化过的本地变量和 for 循环中的本地变量。

它其实是个语法糖，在语义上并没有任何变化。不过，该特性有可能在 Java 开发者当中引起热议。

其他三个变更都将在性能方面带来一些影响。

JEP 307 解决了 G1 垃圾回收器的一个问题——截止到 Java 9，G1 的 Full GC 采用的是单线程算法。也就是说，G1 在发生 Full GC 时会严重影响性能。JEP 307 的目的就是要采用并行 GC 算法，在发生 Full GC 时可以使用多个线程进行并行回收。

JEP 310 对类数据共享 (CDS) 进行了扩展，JVM 可以将一些类记录到一个共享的压缩文件里，在 JVM 下一次启动时可以将这个文件映射到 JVM 进程，以此来减少启动时间。该文件也可以在多个 JVM 间共享，在

同一个机器上运行多个 JVM 时，这样做可以减少内存占用。

该功能在 Java 5 中就已存在，但截止到 Java 9，该功能只允许 bootstrap 类加载器加载压缩的类。JEP 310 的目的是扩展该功能，让应用程序和自定义类加载器也能加载压缩的类。该特性目前仅在 Oracle JDK 中可用，OpenJDK 并不包含该特性。

JEP 计划将该特性从 Oracle 私有仓库中迁移到公共仓库，从 Java 10 往后，常规版本（非 LTS）将会使用 OpenJDK 的二进制包。此举表明有用户正在使用该特性，所以需要在 OpenJDK 中也支持该特性。

JEP 312 旨在改进虚拟机性能，在应用程序线程上调用回调不再需要执行全局虚拟机安全点操作，这意味着 JVM 可以停止单个线程。一些底层小改进包括：

- 降低堆栈跟踪取样所带来的影响（如进行profiling）。
- 减少信号依赖以获得更好的堆栈取样。
- 通过停止单独线程改进偏向锁。
- 从JVM移除了一些内存屏障。

从整体来看，Java 10 似乎并没有包含重大新特性或性能改进。这是可以理解的，毕竟这是新发布周期下的第一个版本。

阿里重启维护 Dubbo 了

作者 雨多田光



2012 年，阿里巴巴在 GitHub 上开源了基于 Java 的分布式服务治理框架 Dubbo，之后它成为了国内该类开源项目的佼佼者，许多开发者对其表示青睐，同时，先后有不少公司在实践中基于 Dubbo 进行分布式系统架构。目前在 GitHub 上，它的 fork、star 数均已破万。

今年 9 月底，同为阿里开源的项目 RocketMQ 被 Apache 社区接纳为顶级项目；10 月中旬，OpenMessaging、ApsaraCache 等全球化开源项目在阿里云栖大会正式公布；同时，Dubbo 也被列入重点维护开源项目，期望继续保持快速发展的态势。

阿里这一系列开源项目的成绩，让人们看到了它对于开源这件事的

重视，重启维护 Dubbo 的背后，有什么样的思考呢？InfoQ 就此采访了 Dubbo 负责人、阿里巴巴中间件高级技术专家罗毅。

InfoQ：为什么重启维护？有哪些因素在驱动着？

罗毅：Dubbo 自开源以来，深受国内友商和开源爱好者的青睐，虽然一直陆续在维护，但是由于 Dubbo 用户群体庞大，日常维护根本无法完全满足社区的旺盛需求。随着集团内部技术水平的迅速发展，如今不仅能够保证集团及客户的系统高效运行，还能抽调更多精力将技术赋能给全社会。

开源就是阿里巴巴集团在技术层面赋能的重要领域。阿里巴巴中间件团队今后不仅要聆听社区的声音，及时修复问题，及时合并优秀的 pull request，还会力争将 Dubbo 打造成有国际影响力的 RPC 框架。

从集团层面看，阿里为国内甚至国际开源社区贡献了大量优秀开源项目，如大家熟知的 RocketMQ、JStorm、Fastjson、Dubbo、Weex 等。在今年的云栖大会上，阿里集团公开宣布了将加大技术投入、拥抱开源的发展策略。正是由于以上几个原因，阿里巴巴中间件团队决定 Dubbo 的下一步计划是持续发展，并走向国际化。

InfoQ：我们知道阿里内部现在基本上没有在使用 Dubbo，而是用了 Dubbo 之后开发的第三代 RPC 服务框架 HSF（High-speed Service Framework），那现在还将 Dubbo 重启维护，大家不免疑惑。

罗毅：Dubbo 和 HSF 都是阿里巴巴集团自研的 RPC 服务框架，在不同时期都很好的支持了集团业务的发展。目前，HSF2 主要服务于集团内部业务，而 Dubbo2 主要以开源的形式服务社会，它们之间的关系与 Google 内部使用的 Stubby 和开源的 gRPC 类似。

从功能及使用方式上来说，HSF2 和 Dubbo2 都是十分优秀的 RPC 框架，都能很好地满足搭建分布式服务化系统的诉求。内部坚持使用 HSF2 而不是开源版本的 Dubbo2 与业务属性和规模有关。

此外也出于以下几方面的考虑：内部统一技术框架、统一运维方面的诉求，应用迁移成本，内部服务注册发现、配置推送以及链路追踪等外围

系统的集成度等。总的来说，HSF2 在服务治理、超大规模集群、多机房几个方面比 Dubbo2 更有优势，并且在使用层面保持了对 Dubbo2 的兼容性。

为什么我们还要重启 Dubbo 的维护呢？

道理和 Google 开源 gRPC 是一样的。开源不仅仅是赋能社会的方式，我们也可以通过社区反馈提升产品和技术能力。

Dubbo2 作为一款优秀的开源产品，由于面向的用户群体非常广泛，这就决定了它的设计原则强调扩展性、使用轻量、以及对开源外围系统和协议的适配。通过开源社区的建议，目前 Dubbo 已经具备了一些特有功能，例如对 REST 的支持和对 Spring Boot 的集成。

值得注意的是，目前负责 Dubbo 的团队和内部负责 HSF 的是同一个团队，在聆听外部用户反馈之余，我们也会把大规模领域里的服务运维经验反哺回 Dubbo 社区，形成良性循环，做到真正意义上的内外统一。

InfoQ：那这么多年过去了，现在 Dubbo 在同类型项目中还存在优势吗？

罗毅：Dubbo 目前在 GitHub 上有超过 12000 个 star 和超过 10000 的 fork 数，仍然是国内影响力最大的开源项目之一。这其中有两个重要因素，一个是 RPC 领域相对成熟，自 Dubbo 开源的第一天起，框架里已经融合了阿里巴巴服务化改造进程中沉淀下来的诸多宝贵经验。

其二是 Dubbo 设计上十分提倡可扩展性，在框架内置功能不能满足业务诉求甚至过时的情况下，用户可以选择自行扩展。这一点，从友商给我们提交来的 pull request 中可以明显感受到，优秀的框架设计本身就可以很好的支持用户千变万化的需求。

Dubbo 维护重启后，3 个月内连续发布 3 个维护版本，不仅修复了优雅停机、注解配置等一些框架缺陷，还新增了 Netty4 通信模块和线程堆栈 dump 特性等。在框架稳定性上已经有了大幅提升，今后 Dubbo 将持续保持快速迭代更新，以满足用户的各种需求。

InfoQ：如您所说，这个开源项目，很多代码贡献者和框架采用者其

实提交了不少有意义的反馈，近期 Dubbo 官方一连给出的这三个版本更新，其中意义最大、大家最想看到的更新内容是什么呢？

罗毅：的确，作为国内使用面最广的服务框架的代表，很多用户都贡献了在使用过程中发现的问题和建议。我们目前优先级最高的任务就是在其中遴选社区关注度最高的问题和建议优先给予支持，其中包括 issue 的修复、第三方依赖的升级、新技术新规范的适配、以及功能上的优化等。

已经发布的版本中我们筛选了社区反馈的框架缺陷、pull request 等，按照优先级顺序修复，保证框架的可用性和稳定性，并同时更新了主页 (<http://dubbo.io/>) 和文档 (<https://www.gitbook.com/@dubbo>)。最近即将发布的版本中完善了注解形式的配置、Docker 环境中部署的问题等几个社区呼声较高的需求。

目前团队最大的任务就是活跃社区，聆听用户的声音，已经发布的三个版本以及未来的几个维护版本都是围绕这个话题进行。

InfoQ：阿里对 Dubbo 接下来的发展有怎样的计划？能否给大家一个清晰的视图？

罗毅：主力开发以阿里巴巴中间件团队为主，优先吸纳集团内部对 Dubbo 开源有热情的开发同学，同时积极与国内大量使用 Dubbo 框架的友商联系。一方面是合并大家的建议，对呼声最高的建议进行性能提升，另一方面是寻求共建开源项目的资源。总的来说，项目方面会以阿里内部专门的团队为主，并积极发展社区中的 committer。

在活跃社区的前提下，我们会继续在 Dubbo 框架现代化、国际化这两个大的方向上进行探索。现代化方面主要是考虑到目前微服务架构以及容器化日渐流行的大趋势，Dubbo 作为 RPC 框架如何很好地融入其中，成为其生态体系中不可或缺的一个组件。

这里就不得不提到目前的一些文章在谈到微服务的时候总是拿 Spring Cloud 和 Dubbo 来对比，需要强调的是 Dubbo 未来的定位并不是要成为一个微服务的全面解决方案，而是专注在 RPC 领域，成为微服务生态体系中的一个重要组件。至于大家关注的微服务化衍生出的服务治理需求，

我们会在 Dubbo 积极适配开源解决方案，甚至启动独立的开源项目予以支持。

对于国际化方面的思考是虽然 Dubbo 在 GitHub 上非常受欢迎，但是受众主要来自国内各友商以及个人开发者，希望将来能够将用户拓展到全球，代表国人在 RPC 领域与 gRPC、Finagle 等竞争。

最后，感谢 InfoQ 对 Dubbo 开源项目的关注。有关 Dubbo 项目今后的发展，还请密切关注 GitHub 上的 [Release Notes](#) 以及 Dubbo 的云栖社区专栏。

Linus 怒喷谷歌安全工程师

作者 覃云



11月中旬，Linux之父Linus Torvalds在内核邮件列表上用很犀利的言辞抨击了Google Pixel安全团队的开发者Kees Cook，引起了大家广泛地讨论。

事情的起因在于Google Pixel安全团队的开发者Kees Cook向Linus递交了加固usercopy的pull request，但是Linus Torvalds认为这种请求是极其愚蠢的，因为他认为此类的加固触及到了Linux的核心，会导致内核出现混乱。而且他认为安全人员的很多行为都是让人难以接受的，解决安全问题的核心在于调试和修复bug，而不是应该像安全人员那样靠杀死机器或终止运行来解决问题。

对此，Rober Graham 对 Linus torvalds 的言论表示赞同，他认为我们应该关注邮件的中心思想而不是激烈的言辞，他表示 Linus Torvalds 在邮件中要表达的意思有两点：

对内核进行大的改动应该在小的迭代步骤中进行，而且每一次都应该彻底调试；

次要的安全问题不是重大的紧急情况，他们不允许绕过的规则比 bug 或功能多。

去年曾经有一些安全固化的代码被添加到内核中，以防止一类缓冲区溢出 / 越界的问题，此代码没有解决任何特定 0day 的漏洞，但它能预防一类未来的潜在漏洞，这个代码可以说是有 bug 的，但是不能说它是罪恶的，因为所有的代码都会有 bug。

在 Linus Torvalds 看来，当检测到溢出 / 越界访问时，代码将终止用户模式进程或内核，那么可以说这个代码罪恶的。Linus 认为它应只产生警告，让有问题的代码继续运行。但杀死这些东西将会使得 bug 变得更加糟糕，它会导致内核的灾难性故障，如果我们的车上运行着 Linux 的多个副本，那么这种灾难将会危及我们的生命，而警告虽然会把这些 bug 显示出来，但不会造成灾难性的后果。在经过仅仅一年之后，当 bug 得到修复时，代码的默认行为会被改变并消除错误的代码，从而防止 bug 被利用。

简而言之，在内核中进行大的改变应该在小而可管理的步骤中进行，固化代码在 Linux 的 25 年历史中都没有出现过，所以在非紧急的情况下，没必要立即进行，更不用说绕过 Linus 提出的开发流程了。

再者，大多数的安全人员不是开发人员，他们实际上并不知道很多事情是如何运行的，边界检查被他们定义为一种用来防止缓冲区溢出的安全功能，但实际上它是一种调试功能，开发人员都知道这一点，但是安全专家往往不知道，而做出这些内核变化的往往是不懂这一点的安全人员，他们没有意识到内核的变化会在现有的代码产生大量的 bug，而且杀死错误代码也是极其不恰当的行为。

由此可见，虽然 Linus 的语气有点不友善，但是他的说法是合理的，

他是一个讲道理的人，他并没有试图阻止对内核的改变，也并没有阻止在安全上的提升，他只是想告诉人们，对内核进行大的改动需要用传统的方式，而不是采取一刀切的方法，与功能和 bug 相比，安全的地位并没有比他们高。

参考链接

- <http://lkml.iu.edu/hypermail/linux/kernel/1711.2/01701.html>
- <http://blog.erratasec.com/2017/11/why-linus-is-right-as-usual.html#.WhPPN7T1XOS>

阿里巴巴正式开源其自研容器技术 Pouch

作者 孙宏亮



11月19日上午，在中国开源年会现场，阿里巴巴正式开源了基于Apache 2.0 协议的容器技术 Pouch。Pouch 是一款轻量级的容器技术，拥有快速高效、可移植性高、资源占用少等特性，主要帮助阿里更快的做到内部业务的交付，同时提高超大规模下数据中心的物理资源利用率。开源之后，Pouch 成为一项普惠技术，人人都可以在 GitHub 上获取，[GitHub 项目地址](#)。

Pouch 的开源，是阿里看好容器技术的一个信号。时至今日，全球范围内，容器技术在大多数企业中落地，已然成为一种共识。如何做好容器的技术选型，如何让容器技术可控，相信是每一个企业必须考虑的问题。

Pouch 无疑使得容器生态再添利器，在全球巨头垄断的容器开源生态中，为中国技术赢得了一块阵地。

Pouch 技术现状

此次开源 Pouch，相信行业中很多专家都会对阿里目前的容器技术感兴趣。到底阿里玩容器是一个侠之大者，还是后起之秀呢？以过去看未来，技术领域尤其如此，技术的沉淀与积累，大致可以看清一家公司的技术实力。

Pouch 演进

追溯 Pouch 的历史，我们会发现 Pouch 起源于 2011 年。当时，Linux 内核之上的 namespace、cgroup 等技术开始成熟，LXC 等工具也在同时期诞生不久。阿里巴巴作为一家技术公司，即基于 LXC 研发了容器技术 t4，并在当时以产品形态给集团内部提供服务。此举被视为阿里对容器技术的第一次探索，也为阿里的容器技术积淀了最初的经验。随着时间的推移，两年后，Docker 横空出世，其镜像技术层面，极大程度上解决了困扰行业多年的“软件封装”问题。镜像技术流行开来后，阿里没有理由不去融合这个给行业带来巨大价值的技术。于是，在 2015 年，t4 在自身容器技术的基础上，逐渐吸收社区中的 Docker 镜像技术，慢慢演变，打磨为 Pouch。

带有镜像创新的容器技术，似一阵飓风，所到之处，国内外无不叫好，阿里巴巴不外如是。2015 年末始，阿里巴巴集团内部在基础设施层面也在悄然发生变化。原因很多，其中最简单的一条，相信大家也不难理解，阿里巴巴体量的互联网公司，背后必定有巨大的数据中心在支撑，业务的爆炸式增长，必定导致基础设施需求的增长，也就造成基础设施成本的大幅提高。容器的轻量级与资源消耗低，加上镜像的快速分发，迅速让阿里巴巴下定决心，在容器技术领域加大投入，帮助数据中心全面升级。

阿里容器规模

经过两年多的投入，阿里容器技术 Pouch 已经在集团基础技术中，扮演着极其重要的角色。2017 年双 11，巨额交易 1682 亿背后，Pouch 在“超级工程”中做到了：

- 100% 的在线业务 Pouch 化
- 容器规模达到百万级

回到阿里集团内部，Pouch 的日常服务已经覆盖绝大部分的事业部，覆盖的业务场景包括：电商、广告、搜索等；覆盖技术栈包括：电商应用、数据库、大数据、流计算等；覆盖编程语言：Java、C++、NodeJS 等。

Pouch 技术优势

阿里巴巴容器技术如此之广的应用范围，对行业来说实属一大幸事，因为阿里已经用事实说明：容器技术已经在大规模生产环境下得到验证。然而，由于 Pouch 源自阿里，而非社区，因此在容器效果、技术实现等方面，两套体系存在差异。换言之，Pouch 存在不少独有的技术优势。

隔离性强

隔离性是企业走云化之路过程中，无法回避的一个技术难题。隔离性强，意味着技术具备了商用的初步条件；反之则几乎没有可能在业务线上铺开。哪怕是阿里巴巴这样的技术公司，实践容器技术伊始，安全问题都无法幸免。众所周知，行业中的容器方案大多基于 Linux 内核提供的 cgroup 和 namespace 来实现隔离，然后这样的轻量级方案存在弊端：

- 容器间，容器与宿主间，共享同一个内核；
- 内核实现的隔离资源，维度不足。

面对如此的内核现状，阿里巴巴采取了三个方面的工作，来解决容器的安全问题：

- 用户态增强容器的隔离维度，比如网络带宽、磁盘使用量等；
- 给内核提交 patch，修复容器的资源可见性问题，cgroup 方面的 bug；

- 实现基于 Hypervisor 的容器，通过创建新内核来实现容器隔离。

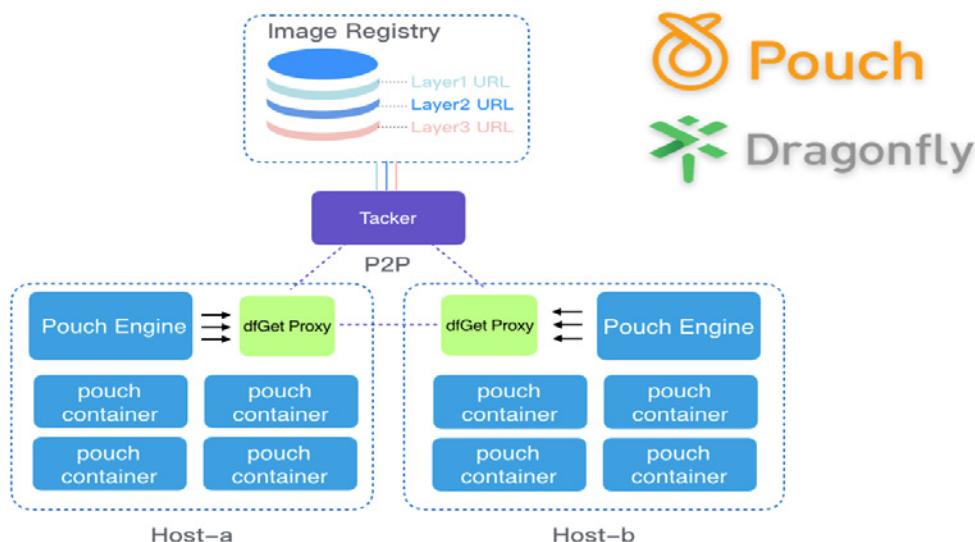
容器安全的研究，在行业中将会持续相当长时间。而阿里在开源 Pouch 中，将在原有的安全基础上，继续融合 lxcfs 等特性与社区共享。同时阿里巴巴也在计划开源“阿里内核”，将多年来阿里对 Linux 内核的增强回馈行业。

P2P 镜像分发

随着阿里业务爆炸式增长，以及 2015 年之后容器技术的迅速普及，阿里容器镜像的分发也同时成为亟待解决的问题。虽然，容器镜像已经帮助企业在应用文件复用等方面，相较传统方法做了很多优化，但是在数以万计的集群规模下，分发效率依然令人抓狂。举一个简单例子：如果数据中心中有 10000 台物理节点，每个节点同时向镜像仓库发起镜像下载，镜像仓库所在机器的网络压力，CPU 压力可想而知。

基于以上场景，阿里巴巴镜像分发工具“蜻蜓”应运而生。蜻蜓是基于智能 P2P 技术的通用文件分发系统。解决了大规模文件分发场景下分发耗时、成功率低、带宽浪费等难题。大幅提升发布部署、数据预热、大规模容器镜像分发等业务能力。目前，“蜻蜓”和 Pouch 同时开源，[项目地址](#)。

Pouch 与蜻蜓的使用架构图如下：



富容器技术

阿里巴巴集团内部囊括了各式各样的业务场景，几乎每种场景都对 Pouch 有着自己的要求。如果使用外界“单容器单进程”的方案，在业务部门推行容器化存在令人难以置信的阻力。阿里巴巴内部，基础技术起着巨大的支撑作用，需要每时每刻都更好的支撑业务的运行。当业务运行时，技术几乎很难做到让业务去做改变，反过来适配自己。因此，一种对应用开发、应用运维都没有侵入性的容器技术，才有可能大规模的迅速铺开。否则的话，容器化过程中，一方面得不到业务方的支持，另一方面也需要投入大量人力帮助业务方，非标准化的实现业务运维。

阿里深谙此道，内部的 Pouch 技术可以说对业务没有任何的侵入性，也正是因为这一点在集团内部做到 100% 容器化。这样的容器技术，被无数阿里人称为“富容器”。

“富容器”技术的实现，主要是为了在 Linux 内核上创建一个与虚拟机体验完全一致的容器。如此一来，比一般容器要功能强大，内部有完整的 init 进程，以及业务应用需要的任何服务，当然这也印证了 Pouch 为什么可以做到对应用没有“侵入性”。技术的实现过程中，Pouch 需要将容器的执行入口定义为 systemd，而在内核态，Pouch 引入了 cgroup namespace 这一最新的内核 patch，满足 systemd 在富容器模式的隔离性。从企业运维流程来看，富容器同样优势明显。它可以在应用的 Entrypoint 启动之前做一些事情，比如统一要做一些安全相关的事情，运维相关的 agent 拉起。这些需要统一做的事情，倘若放到用户的启动脚本，或镜像中就对用户的应用诞生了侵入性，而富容器可以透明的处理掉这些事情。

内核兼容性

容器技术的井喷式发展，使得不少走在技术前沿的企业享受到技术红利。然后，“长尾效应”也注定技术演进存在漫长周期。Pouch 的发展也在规模化进程中遇到相同问题。

但凡规模达到一定量，“摩尔定律”决定了数据中心会存有遗留资源，

如何利用与处理这些物理资源，是一个大问题。阿里集团内部也是如此，不管是不同型号的机器，还是从 2.6.32 到 3.10+ 的 Linux 内核，异构现象依然存在。倘若要使所有应用运行 Pouch 之中，Pouch 就必须支持所有内核版本，而现有的容器技术支持的 Linux 内核都在 3.10 以上。不过技术层面万幸的是，对 2.6.32 等老版本内核而言，namespace 的支持仅仅缺失 user namespace；其他 namespace 以及常用的 cgroup 子系统均存在；但是 /proc/self/ns 等用来记录 namespace 的辅助文件当时还不存在，setns 等系统调用也需要在高版本内核中才能支持。而阿里的技术策略是，通过一些其他的方法，来绕过某些系统调用，实现老版本内核的容器支持。

当然，从另一个角度而言，富容器技术也很大程度上，对老版本内核上的其他运维系统、监控系统、用户使用习惯等实现了适配，保障 Pouch 在内核兼容性方面的高可用性。

因此综合来看，在 Pouch 的技术优势之上，我们不难发现适用 Pouch 的应用场景：传统 IT 架构的迅速容器化，企业大规模业务的部署，安全隔离要求高稳定性要求高的金融场景等。

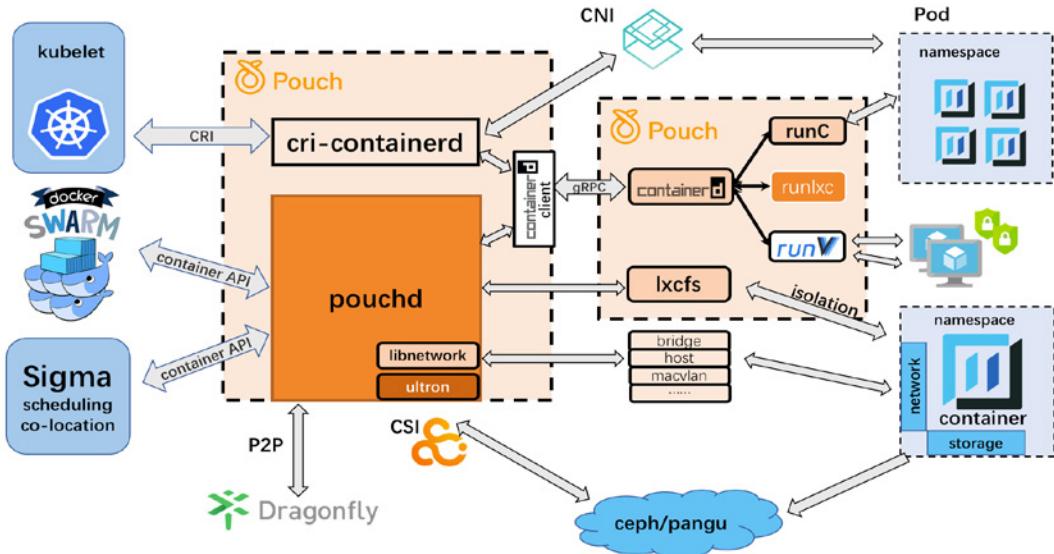
Pouch 架构

凭借差异化的技术优势，Pouch 在阿里巴巴大规模的应用场景下已经得到很好的验证。然而，不得不说的是：目前阿里巴巴内部的 Pouch 与当前开源版本依然存在一定的差异。

虽然优势明显，但是如果把内部的 Pouch 直接开源，这几乎是一件不可能的事。多年的发展，内部 Pouch 在服务业务的同时，存在与阿里内部基础设施、业务场景耦合的情况。耦合的内容，对于行业来说通用性并不强，同时涉及一些其他问题。因此，Pouch 开源过程中，第一要务即解耦内部依赖，把最核心的、对社区同样有巨大价值的部分开源出来。同时，阿里希望在开源的最开始，即与社区站在一起，共建 Pouch 的开源社区。随后，以开源版本的 Pouch 逐渐替换阿里巴巴集团内部的 Pouch，最终达成 Pouch 内外一致的目标。当然，在这过程中，内部 Pouch 的解耦工作，

以及插件化演进工作同样重要。而在 Pouch 的开源计划中，明年 3 月底会是一个重要的时间点，彼时 Pouch 的 1.0 版本将发布。

从计划开源的第一刻开始，Pouch 在生态中的架构图就设计如下：



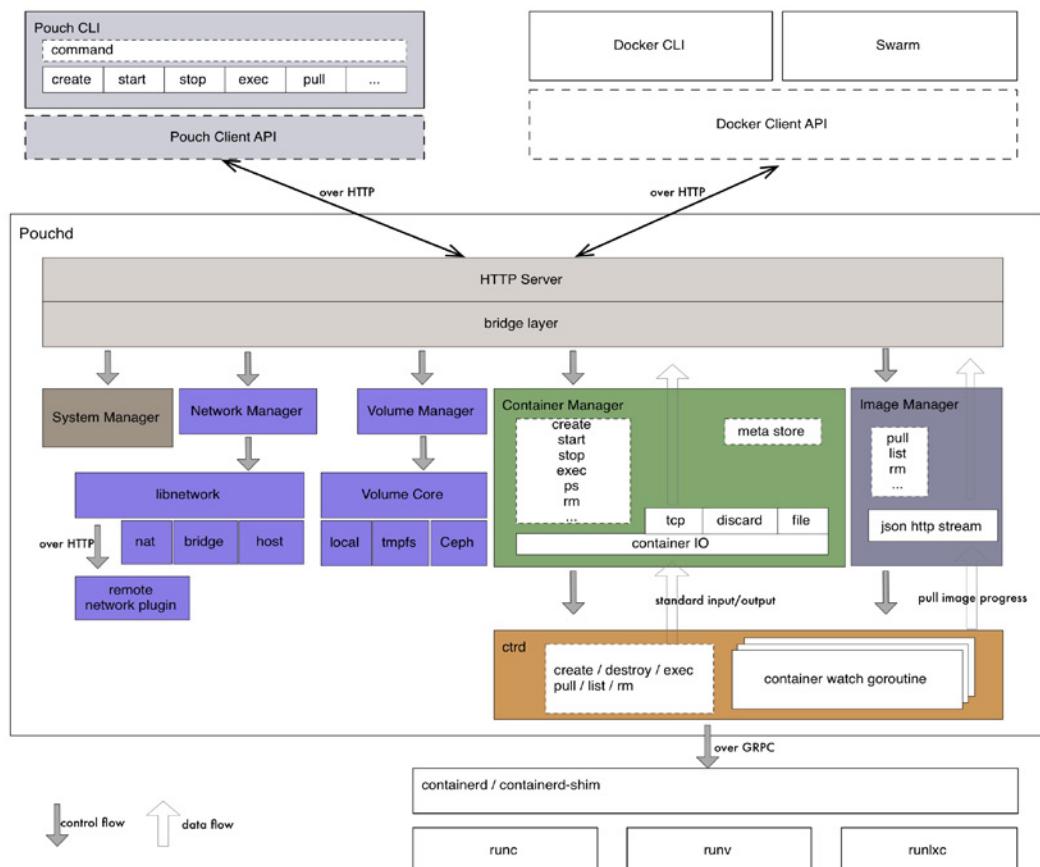
Pouch 的生态架构可以从两个方面来看：第一，如何对接容器编排系统；第二，如何加强容器运行时。

容器编排系统的支持，是 Pouch 开源计划的重要板块。因此，设计之初，Pouch 就希望自身可以原生支持 Kubernetes 等编排系统。为实现这点，Pouch 在行业中率先支持 container 1.0.0。目前行业中的容器方案 **containerd** 主要停留在 0.2.3 版本，新版本的安全等功能还无法使用，而 Pouch 已经是第一个吃螃蟹的人。当前 Docker 依然是 Kubernetes 体系中较火的容器引擎方案，而 Kubernetes 在 runtime 层面的战略计划为使用 **cri-containerd** 降低自身与商业产品的耦合度，而走兼容社区方案的道路，比如 **cri-containerd** 以及 **containerd** 社区版。另外，需要额外提及的是，内部的 Pouch 是阿里巴巴调度系统 Sigma 的重要组成部分，同时支撑着“混部”工程的实现。Pouch 开源路线中，同样会以支持“混部”为目标。未来，Sigma 的调度（scheduling）以及混部（co-location）能力有望服务行业。

生态方面，Pouch 立足开放；容器运行时方面，Pouch 主张“丰富”与“安全”。**runC** 的支持，可谓顺其自然。**runV** 的支持，则表现出了和生态的差

异性。虽然 docker 默认支持 runV，然而在 docker 的 API 中并非做到对“容器”与“虚拟机”的兼容，从而 Docker 并非是一个统一的管理入口。而据我们所知，现有企业中仍有众多存量虚拟机的场景，因此，在迎接容器时代时，如何通过统一的运维入口，同时管理容器和虚拟机，势必会是“虚拟机迈向容器”这个变迁过渡期中，企业最为关心的方案之一。Pouch 的开源形态，很好的覆盖了这一场景。runlxc 是阿里巴巴自研的 lxc 容器运行时，Pouch 对其的支持同时也意味着 runlxc 会在不久后开源，覆盖企业内部拥有大量低版本 Linux 内核的场景。

Pouch 对接生态的架构如下，而 Pouch 内部自身的架构可参考下图：



和传统的容器引擎方案相似，Pouch 也呈现出 C/S 的软件架构。命令行 CLI 层面，可以同时支持 Pouch CLI 以及 Docker CLI。对接容器 runtime，Pouch 内部通过 container client 通过 gRPC 调用 containerd。

Pouch Daemon 的内部采取组件化的设计理念，抽离出相应的 System Manager、Container Manager、Image Manager、Network Manager、Volume Manager 提供统一化的对象管理方案。

写在最后

如今 Pouch 的开源，意味着阿里积累的容器技术将走出阿里，面向行业。而 Pouch 的技术优势，决定了其自身会以一个差异化的容器解决方案，供用户选择。企业在走云化之路，拥抱云原生（Cloud Native）时，Pouch 致力于成为一款强有力的软件，帮助企业的数字化转型做到最稳定的支持。

Pouch 目前已经在 GitHub 上开源，欢迎任何形式的开源参与。
[GitHub 地址](#)。

作者介绍

孙宏亮，阿里巴巴技术专家，毕业于浙江大学，目前在阿里巴巴负责容器项目 Pouch 的开源建设。数年来一直从事云计算领域，是国内第一批研究和实践容器技术的工程师，在国内起到极为重要的容器技术布道作用。拥有著作《Docker 源码分析》，个人崇尚开源精神，同时是 Docker Swarm 项目的全球 Maintainer。

Kafka 不只是个消息系统

作者 Jay Kreps 译者 薛命灯



Confluent 联合创始人兼 CEO Jay Kreps 发表了一篇博文，给出了 Kafka 的真正定位——它不只是个消息系统，它还是个存储系统，而它的终极目标是要让流式处理成为现代企业的主流开发范式。以下内容翻译自作者的博文，查看原文 [It's Okay To Store Data In Apache Kafka](#)。

人们总是问是否可以把 Kafka 作为长期的数据存储来使用，很显然，如果把数据保留策略设置为“永久”或者启用主题的日志压缩功能，那么数据就可以被永久保存下来。但我觉得人们其实真正想知道的是，这样做是不是很疯狂。

简而言之，这样做不算疯狂。实际上，人们一直都在这么做，而且 Kafka 的设计意图之一就是要将它作为数据存储系统。不过问题是，为什么我们要把 Kafka 作为数据存储呢？

1. 你可能在构建一个基于事件溯源的应用程序，需要一个数据存储来保存变更日志。理论上，你可以使用任何一种存储系统。Kafka 已经解决了不可变（immutable）日志和基于这些日志生成“物化视图”的问题，既然这样，为什么不直接使用 Kafka 呢？纽约时报已经在他们的 CMS 系统里使用 Kafka 来保存他们的文章。
2. 你可能在应用程序里使用了缓存，并从 Kafka 上获取数据来更新缓存。你可以将 Kafka 的主题设置为压缩型日志，应用程序每次在重启时就可以从零偏移量位置重新刷新缓存。
3. 你的流式作业数据流来自 Kafka，在流式作业的逻辑发生变更后，需要重新计算结果。最简单的办法就是将偏移量重置为零，让新代码重新计算结果。
4. Kafka 经常被用于捕获和分发数据库的变更事件（通常被称为 CDC，Change Data Capture）。应用程序可能只需要最新的数据库变更，但却要处理完整的数据快照，而这是相当耗时的操作。如果启用主题的日志压缩功能，就可以让应用程序直接从零偏移量位置重新加载数据。

像这样在 Kafka 里存储数据并不是什么疯狂事，Kafka 本来就是设计用来存储数据的。数据经过校验后被持久化在磁盘上，并通过复制副本提升容错能力。再多的数据都不会拖慢 Kafka，在生产环境中，有些 Kafka 集群甚至已经保存超过 1 TB 的数据。

那么人们为什么会对使用 Kafka 来存储数据心存疑问呢？

我想，人们更多的是把 Kafka 当成了消息队列系统。消息队列有一些不成文的规则，比如“不要在消息队列里保存消息”。传统的消息系统之所以不能用来保存消息，是因为：

- 消息被读取后就会被删除

- 伸缩性差
- 缺乏健壮的复制机制（如果 broker 崩溃，数据也就丢失了）

传统的消息系统在设计上存在很多不足。从根本上讲，任何一个异步消息系统都会保存消息，只是时间很短，有时候只有几秒钟，直到消息被消费为止。假设有一个服务向消息队列发送消息，并希望有一种机制可以保证其他服务能够收到这个消息，那么消息就需要被保存在某个地方，直到其他服务读取它。如果消息系统不擅长存储消息，也就谈不上给消息“排队”了。你可能觉得无所谓，因为你并不打算长时间地保留消息。但不管怎样，如果消息系统持续地处理负载，总会有一些未被消费的消息需要保存下来。一旦消息系统发生崩溃，如果没有有效的容错存储机制，数据就会丢失。消息存储是消息系统的基础，但人们总是忽略这一点。

实际上，Kafka 并非传统意义上的消息队列，它与 RabbitMQ 等消息系统并不一样。它更像是一个分布式的文件系统或数据库。Kafka 与传统消息系统之间有三个关键区别。

- Kafka 持久化日志，这些日志可以被重复读取和无限期保留。
- Kafka 是一个分布式系统：它以集群的方式运行，可以灵活伸缩，在内部通过复制数据提升容错能力和高可用性。
- Kafka 支持实时的流式处理。

以上三点足以将 Kafka 与传统的消息队列区别开，我们甚至可以把它看成是流式处理平台。

我们可以这样来看待消息系统、存储系统和 Kafka 之间的关系。消息系统传播的是“未来”的消息：你连接到 broker 上，并等待新消息的到来。存储系统保存的是过去写入的数据：你查询或读取的结果是基于过去所做的更新。而流式处理可以把这两者结合起来，既可以处理过去的数据，也可以处理未来的消息。这也就是为什么 Kafka 的核心就是一个持续的、基于时间排序的日志。它是一种结构化的“文件”，而且从逻辑上看，它没有终点，会一直持续下去。应用程序不需要区分已有的旧数据和即将生成的新数据，它们都存在于一条持续的流中。Kafka 提供了统一的协议和 API

来保存过去的数据和传播未来的消息，Kafka 因此成为一种非常好的流式处理平台。

日志就像是分布式文件系统中的一个文件，在这个系统里，日志被复制到多台机器上，被持久化到磁盘，并支持高吞吐的线性读取和写入。当然，日志也像是一个消息系统，支持高吞吐的并发写入和低延迟的多消费者。

从实现方面来看，日志非常适合用来作为数据存储。Kafka 本身就是使用复制日志作为存储，所以你也不例外！在 Kafka 内部，偏移量被保存在一个压缩主题上，Kafka Streams API 使用压缩主题来记录应用程序的处理状态。

当然，把 Kafka 作为存储系统来用并不会给你带来新的门槛。存储系统包揽了正确性、运行时间和数据完整性等方面的工作。如果一个系统成为数据的标准来源，人们就会对它的正确性和运维标准提出很高的要求。我们花了大量的精力在提升 Kafka 的正确性上，我们每天在数百台机器上运行数个小时的分布式测试以及数千个常规性的单元测试，但我们觉得还有很多事情要做。除了测试之外，我们还需要知道如何做好运维工作，以及了解系统的局限性。

有时候，人们也会问我，这是不是就意味着 Kafka 可以取代其他存储引擎。答案当然是否定的。

首先，数据库提供大量的查询，而 Kafka 并不打算在日志上增加随机访问的特性。Kafka 保存数据可以被复制到其他数据库、缓存、流式处理器、搜索引擎、图存储引擎和数据湖（data lake）上，这些存储引擎都各自的优缺点，我们也无法做出一个可以打败其他所有引擎的系统。

如果说 Kafka 并不想取代这些系统，那它存在的意义是什么？你可以把数据中心看成是一个大型的数据库，Kafka 是这个系统里的提交日志，而其他存储引擎则是索引或视图。Kafka 是构建数据库的基础，至于查询方面的工作可以交给索引和视图。

Kafka Streams API 提供了交互式的查询功能。基于 Kafka Streams 开发的应用就是一个 Kafka 消费者，只不过它们可以维护计算状态，而且这

些状态可以直接保存到外部的存储系统，这种物化视图让 Kafka 具备了低延迟的查询能力。Kafka 集群保存日志，Streams API 保存物化视图并处理查询请求。后来我们引入了 KSQL——Kafka 的流式 SQL 引擎。有了 [KSQL](#)，用户可以直接使用 SQL 语句从 Kafka 上获得物化视图。

我们不打算为 Kafka 提供查询 API 的另一个原因是因为我们有其他更重要的事情要做。我们希望流式处理成为主流的开发模式，让流式平台成为现代数字业务的中心系统。我们希望能够达成这个让人激动不已的目标，而不只是创建一种新的数据库系统。我们相信，在现代企业里，流式平台将会成为移动和处理数据的黑马。要实现这个目标，我们还有很多事情要做。

为什么原生应用开发者需要关注 Flutter

作者 Animesh Jain 译者 薛命灯



Flutter 是由谷歌创建的一个移动应用 SDK，用于构建“现代移动应用”。目前它还处于 alpha 阶段，不过它的文档和相关工具十分齐全，有些移动应用已经在使用 [Flutter](#)。

在这篇文章里，我将分享使用 Flutter 开发一个移动应用的愉快经历，并告诉大家为什么我这么喜欢 Flutter。

背景简介

我开发了一个叫作“[Chips of Fury](#)”的扑克牌游戏（可以从 [Play Store](#) 和

[App Store](#) 下载），如果你想和朋友们玩扑克牌游戏，但手头又没有扑克牌，那么就可以使用这款应用。这是一款多人游戏，要求参与游戏的多个设备之间支持实时同步，而且用到了大量的自定义 UI 元素。

我花了一个半月时间开发这款游戏，其中包括用在学习 Flutter 上的时间。之前也尝试过使用 Android/Java 和 iOS/Swift 来开发，但都无疾而终，主要是因为需要做太多繁琐的工作。而 Flutter 不仅帮助我加快开发速度，还让我坚持到了最后。所以，我现在成了 Flutter 的超级大粉丝！

我认为 Flutter 会成为移动开发的未来，个中缘由且听我慢慢说来。

1. 简单的 Dart

当我跟其他开发者说起 Flutter，他们会很诧异：“什么？ [Dart](#)？”毕竟 Dart 跟其他语言（如 Swift 或 Kotlin）比起来在语法方面并没有什么优势。

但其实 Dart 学起来很容易，根本不会成为入门 Flutter 的阻碍。请看下面的例子。

```
class Vehicle {  
    final int numberOfWorks;  
    final double mileage;  
    final double horsePower;  
  
    int speed = 0;  
  
    Vehicle(this.numberOfWorks, this.mileage, this.horsePower);  
  
    void accelerate() {  
        speed = speed + 1;  
    }  
  
    void decelerate() {  
        speed = speed - 1;  
    }  
    void brake() {
```

```

    speed = 0;
}

}

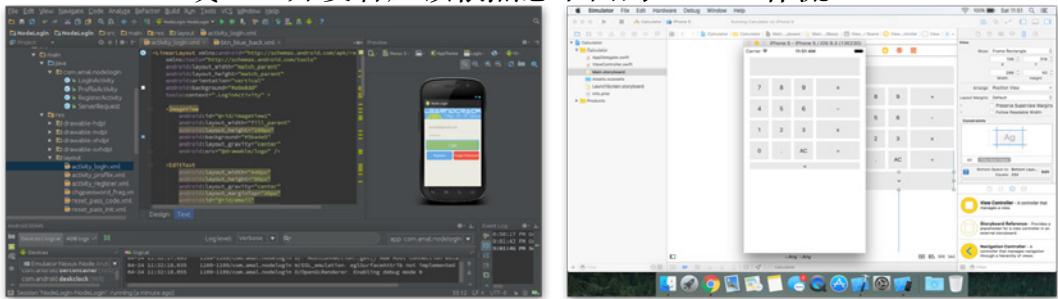
```

可以看出，Dart 的语法与其他面向对象编程语言没有什么太大差别，虽然也存在一些特定的语法，但整体的学习曲线还是很平缓的。

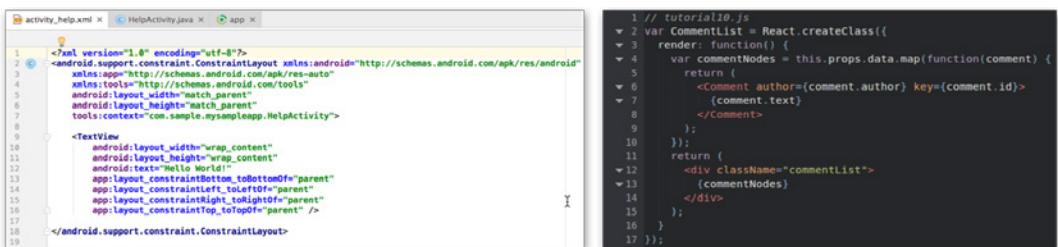
况且，谷歌选择 Dart 也是有原因的。Flutter 需要不断快速地创建和销毁短生命周期对象，Dart 的垃圾回收机制非常适合用于完成这项任务。

2. 高效的 Dart

Android 或 iOS 开发者应该很熟悉下面的 IDE 工作流：



当然，也很熟悉下面的这些标记或代码：



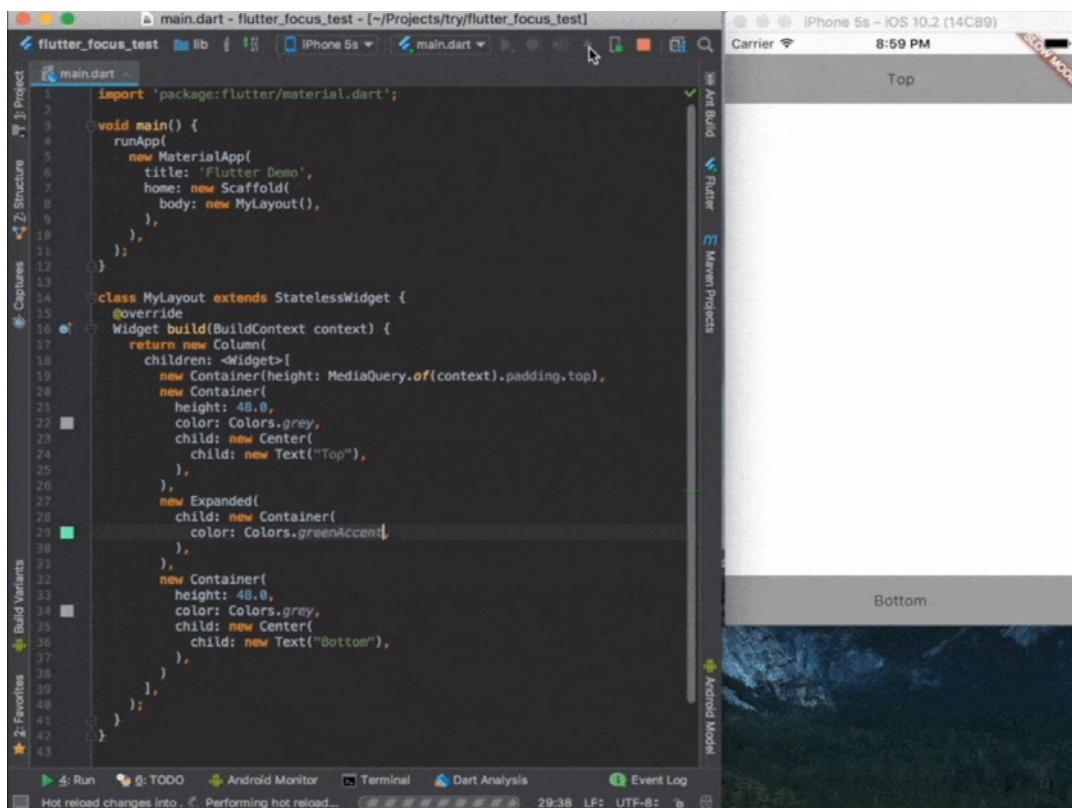
在 Flutter 里，界面布局直接通过 Dart 编码来定义，不需要使用 XML 或模板语言，也不需要使用可视化设计器之类的工具。

说到这里，大家可能会一脸茫然，就像我当初的反应一样。使用可视化工具不是更容易吗？如果把所有的逻辑都写到代码里不是会让事情变复杂吗？

其实不然，请看下面的例子。

首先，Flutter 提供了热加载功能（Hot Reload）。

下图左边是代码，这些代码足以用于运行一个应用。右边是模拟器，自定义了一个布局，包括顶部和底部的栏位以及中间的内容容器。

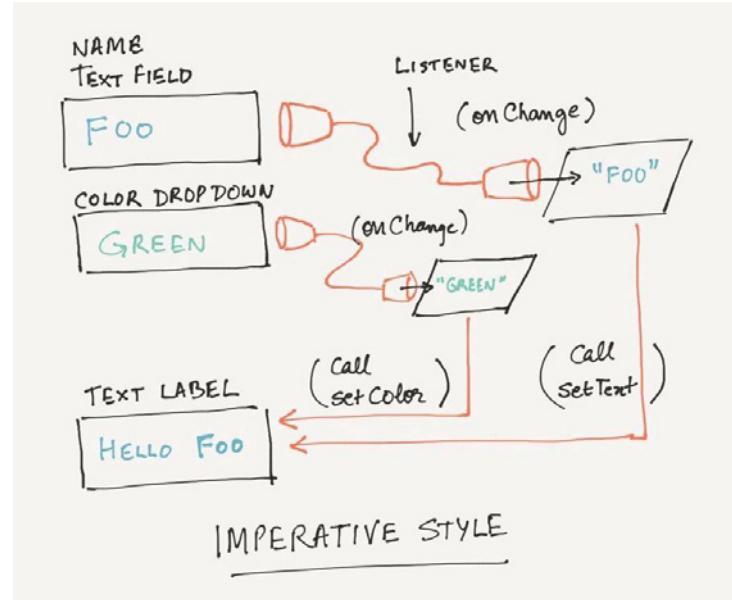


这里不需要布局用的 XML 文件，也没有 xib 等文件。在左边修改代码，可以立即热加载到右边的模拟器上。热加载还能保持应用的状态，所以在热加载之后不需要重新导航到之前的页面。这比 Android 的 Instant Run 不知道要领先多少年。对于大型的应用同样适用。如此快的速度，正是 Dart 的优势所在。

在 Flutter 中进行布局要比在 Android/XCode 中快得多。

3. 反应式的 Flutter

假设要实现下列的应用，如果按照命令式的编程风格，可能是这样实现的：

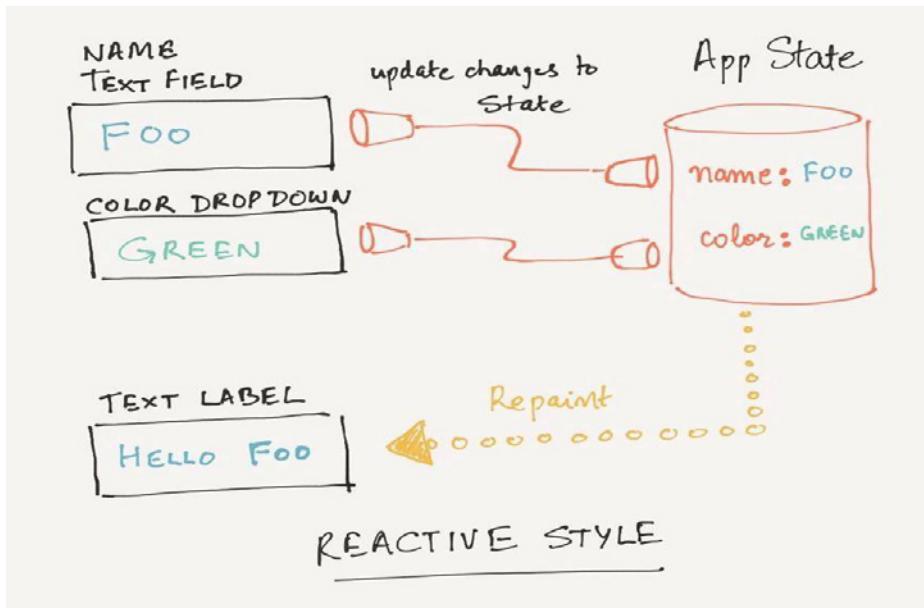


我们需要为文本框和下拉列表添加监听器，在发生事件变更时，先获取文本标签再调用相关的方法进行赋值。

在 Android 或 iOS 的公共 API 里，View 的 setter 和 getter 方法不计其数，光是 TextView 就有 259 个这样的方法，另外还有 4 个构造函数。所以在设置这些部件的属性时真是一件让人头疼的事情。

而反应式的编程风格会这样来实现：

我们为文本框和下拉列表添加监听器，当发生事件变更时，在全局状态”



里更新这些值，并让 Flutter 进行 UI 重绘。Flutter 将自动更新文本标签的值。

4. 一切都是部件 (widget)

在 Android 和 iOS 上，部件所对应的就是各种 View 类。

Flutter 采用了不同的概念，部件不仅仅是结构化的元素。Flutter 的部件架构更多地使用了组合，而不是继承，所以部件架构更加强大和灵活。Flutter 官方文档写道：

部件可以用于定义结构化元素（如按钮或菜单）、样式元素（如字体或颜色）、布局（如空白填充）等。

在 Flutter 里，行为也是部件（如 GestureDetector）。InheritedWidget 可用于进行状态管理，AnimatedWidget 可用于构建动画。

遵循组合大于集成的原则，Flutter 从简单的元部件开始，可以构建出非常复杂的部件。Flutter 的 Container Widget 就是由一系列元部件组成的。

The diagram illustrates the construction of a Container widget from various components. It shows a flow of code snippets where each step adds a new component to the current widget. Arrows point from the annotations to the corresponding code lines.

```
override  
Widget build(BuildContext context) {  
    Widget current = child;  
  
    if (child == null && (constraints == null || !constraints.isTight)) ...  
    if (alignment != null)  
        current = new Align(alignment: alignment, child: current);  
    final EdgeInsetsGeometry effectivePadding = _paddingIncludingDecoration;  
    if (effectivePadding != null)  
        current = new Padding(padding: effectivePadding, child: current);  
    if (decoration != null)  
        current = new DecoratedBox(decoration: decoration, child: current);  
    if (foregroundDecoration != null) {  
        current = new DecoratedBox(  
            decoration: foregroundDecoration,  
            position: DecorationPosition.foreground,  
            child: current  
    );  
    if (constraints != null)  
        current = new ConstrainedBox(constraints: constraints, child: current);  
    if (margin != null)  
        current = new Padding(padding: margin, child: current);  
    if (transform != null)  
        current = new Transform(transform: transform, child: current);  
    return current;  
}
```

A container shows a child wrapped in an Align, wrapped in an Padding, wrapped in a DecoratedBox, wrapped in a DecoratedBox, wrapped in a ConstrainedBox, wrapped in a Margin, wrapped in a Transform!

5. 和 Activity 生命周期管理说再见

我想没有人会喜欢把时间花在 Activity（或 Fragment、

ViewController) 的生命周期管理上。对于我来说，Activity 中 Fragment 的异步数据加载和本地状态管理是一个很大的负担。

但在 Flutter 里，这些东西都消失了。

6. 稳定的 60 帧频

Flutter 的应用被编译成本地代码，所以性能方面不存在问题。事实上，我认为它比 Java 或 Swift 更适合用来开发游戏应用。使用反应式编程风格开发的 UI 代码更加清晰，再加上良好的性能，非常适合用来开发游戏。

来自谷歌的 @wmleler1 写了一篇文章解释为什么 Flutter 的渲染速度会这么快：<https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514>。

为了体验 Flutter 的速度，可以试着安装这款 Flutter 图库应用：<https://play.google.com/store/apps/details?id=io.flutter.gallery>。

iOS 版的需要自己从源代码构建：https://github.com/flutter/flutter/tree/master/examples/flutter_gallery。

7. 同时支持 Android 和 iOS

这个就不用多说了。

8. Flutter 社区

Flutter 处于 alpha 阶段，不过社区已经为它提供很好的支持。[Flutter channel](https://github.com/flutter/flutter_channel) 为新手和有经验的开发者提供了交流渠道，很多问题可以在里面得到快速解答。

Flutter 还很年轻，所以肯定会存在一些局限和已知问题，而且未知的问题也会越来越多。Flutter 开发团队正努力解决这些问题。下面列出了几个已知问题：

目前还不支持内嵌地图。<https://github.com/flutter/flutter/issues/73>
对内嵌视频支持得不好，不过可以使用补丁解决这个问题。<https:///>

github.com/flutter/flutter/pull/12525

无法保存实例状态（Android）。所以，如果应用在后台被终止，应用状态就会丢失。目前还没有简单的办法可以解决这一问题。<https://github.com/flutter/flutter/issues/6827>

商汤科技杨帆：AI 落地的关键是算法闭环

作者 蔡芳芳



人脸识别技术，曾经是反乌托邦的科幻小说中出现的想法，现在可能正在成为中国日常生活的一个特色。

广东深圳已经有了人脸识别抓拍行人闯红灯的示范路口，如果你闯红灯的时候被摄像头拍了下来，下次你再试图闯红灯时，你的脸就会出现在街道旁边的显示屏上，显示屏上还会出现一行字：“人脸识别智能抓拍行人闯红灯”。

人脸识别技术已经成为监视领域最有力的新工具之一，地铁站、机场、海关都在使用这项技术。刷脸取款、刷脸支付、刷脸登机等新应用更是层

出不穷，刷手机的时代仿佛也才到来没多久，刷脸时代已经来势汹汹。

今年 9 月下旬，一段被称为“中国天网”监控视频的视频片段在新浪微博和朋友圈里疯传，视频展示了我国最新实时行人检测识别系统，该系统可以实时监测区分出机动车、非机动车和行人，并能准确识别出机动车和非机动车的种类，以及行人的年龄、性别、穿着。而这个系统的背后，其实是商汤科技的 Sense Video 技术。



主打人脸识别技术的商汤科技成立于 2014 年 10 月，其核心创始人汤晓鸥，同时也是香港中文大学教授，领导着计算机视觉实验室，这一特殊的跨界身份似乎也预示了为何商汤科技未来能够横跨学术和商业两界并取得亮眼成绩。商汤科技目前拥有 140 位博士，2016 年 ImageNet 大规模视觉识别挑战赛中，商汤科技联合香港中文大学一举揽下三项冠军；近日，商汤科技与香港中大 - 商汤科技联合实验室，继以 23 篇论文横扫 CVPR 后，又以 20 篇论文力压群雄称霸 ICCV，在全球顶级视觉学术会议上刮起了一阵中国旋风。而在业界落地方面，商汤科技的产品遍布金融、安防、互联网娱乐、AR、智能手机等多个行业场景，与华为、Qualcomm、中国移动、小米等众多公司都达成了合作。2017 年 7 月，商汤科技获得 4.1 亿美元 B

轮融资，成为史上人工智能最高单笔投资，直到 11 月 2 日旷视科技获得 4.6 亿美元 C 轮融资再度刷新这项纪录。

人脸识别大行其道，不免让人对这项技术及其背后的公司产生了许多好奇。人脸识别技术到底有何门道？它经历了怎样的技术演进历程？各家公司宣传的识别正确率百分之 99 点几后面的小数点真的有区别吗？人脸识别技术在商汤是如何落地的？它带来的安全性问题如何应对？带着这些问题，InfoQ 记者来到了商汤科技（下文统称商汤）在深圳的办公室，对商汤科技联合创始人、副总裁杨帆进行了专访。

商汤到底是一家什么样的公司？

提到商汤，大部分人第一反应就是人脸识别，但人脸识别并不足以定义商汤。

在杨帆看来，商汤是一个坚持人工智能原创技术的平台服务提供商，它利用原创的 AI 技术给不同的行业提供平台化服务、赋能各个行业，让 AI 技术真正地去改变每个行业。“当然目前来说，我们的工作主要集中在人工智能的计算机视觉，也就是图像和视频分析的这个领域。毫无疑问，人脸作为一种非常特殊且具有极高价值的影像标识，会是整个图像视频分析领域中占比重非常大的一部分。但同时商汤还经常给不同行业提供其他解决方案，涵盖范围会远远超过人脸识别。”

计算机视觉技术的发展和突破

深度学习使 CV 真正从学术界走向工业应用

杨帆在计算机视觉技术领域沉浸多年，在微软任职期间，他主要从事计算机视觉、计算机图形学等领域的的新技术孵化工作，包括人脸识别、图像物体识别、人像三维重建等；目前商汤的核心技术也是以人脸识别、智能监控、图像识别等为主。作为主导技术落地的负责人，杨帆笑称自己是给公司的研究员们打下手的，但回忆起计算机视觉技术的发展历程，他表示还是有很大的感触。

上世纪 90 年代末期，有一波所谓的人工智能，或至少是人脸识别的热潮。当时在实验室环境下，人脸识别已经能够达到一个相当不错的结果，但离实际应用还是有比较大的差距。从 2004 年杨帆进入微软实习开始，到 2010、2011 年这段时间内，计算机视觉领域的技术进步一直在持续，但主要还是积累期，整个行业的技术进步相对比较缓慢，基本没有太多新的应用和机会。到了 2011-2012 年，随着硬件设备计算能力的进步，以及各大公司开始具备收集海量数据的能力，深度学习变得越来越实用，给行业带来了巨大的改变，从那之后计算机视觉技术就进入了一个特别高速的快车道。计算机视觉技术从学术界蔓延到了工业界，在各行各业都有了越来越多广泛的应用，这是外因。

从内因角度来讲，这一轮以深度学习为核心的视觉技术，对数据的依赖更强了，核心技术研发能力提高了，而且最终得到的成果普适性也变好了。杨帆回忆道，“我以前在微软做过一些人脸识别的工作，在深度学习出现之前，你做一个算法能够把肤色的问题解决得很好，但它可能对光线的问题就很难适应。假如你想要一个对光线适应很好的算法，它可能对肤色问题又解决不好，它的技术突破是单点性的突破。”

而今天，伴随着海量数据的应用，很多识别技术会变成一种相对通用的方法论，可以以更低的成本、更短的时间，快速迁移到不同的领域上，这其中的价值非常巨大。随着人工智能技术的发展，虽然它难度依然很高，但是它的不可知性和风险已经大大降低，在这种情况下，就会有越来越多的企业愿意投入力量到这些技术的研发中，从而带来更大的价值。

以前只有世界顶尖级别的公司才会成立研究院，去做核心技术研究，比如贝尔实验室、微软等。但是今天你会发现完全不一样，我相信未来整个技术在不同行业的落地，对于整个业界生态会有比较大的改变。

基础研究和应用科研，二者不可偏废

业界曾出现一种批评的声音，称现在很多公司和开发者其实对于深度学习的运作原理并不清楚，只知道应用，却不知其所以然。对此，杨帆也

有自己的看法。

杨帆表示，学术界有两套观念，一套观念说知其然不知其所以然是离经叛道、是不对的。对于这个观念，杨帆表示认可，其实现在已经有很多团队，包括商汤也投入力量在进行更加前沿、更加基础性的科研，“这样的基础科研能够指导我们将来在正确的方向上走得更远。”但杨帆认为，基础研究与应用科研，二者不可偏废，完整的科学体系和持续的方向性指引非常重要，但是实证科学也非常 important，企业最终还是要以技术落地的结果说话。

脱离场景谈识别正确率毫无意义

近几年，很多公司在人脸识别技术上投入了大量的研发并取得了亮眼的成绩，其中识别率一直是各家宣传的重点，今年我们能在各类报道中频繁看到各种 99%、99.4%、99.8%，如何理解这些识别率中小数点后面数字的差距？

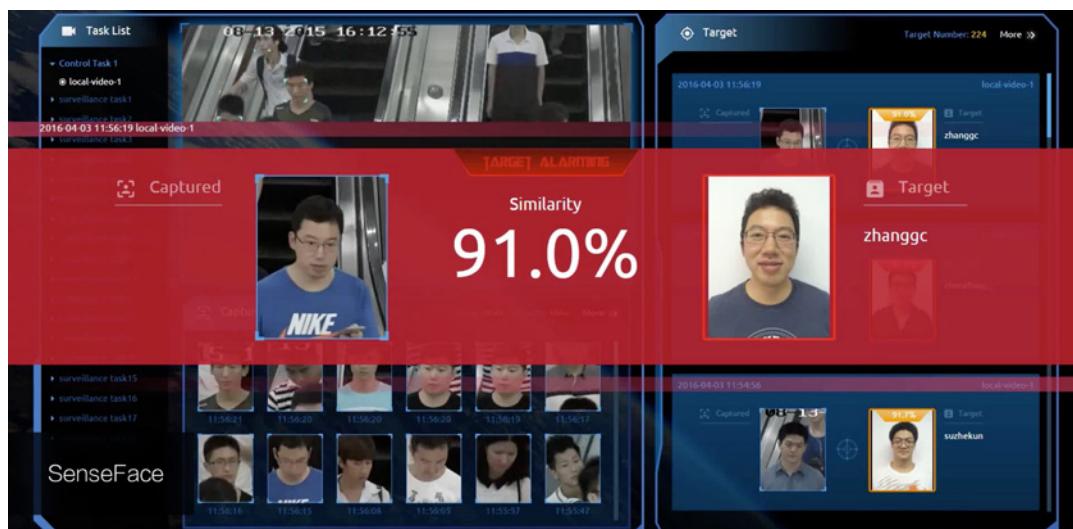
技术指标是没法一概而论的，任何一个技术指标背后都隐藏了一大堆的假设条件。

杨帆列举了几个例子，比如在金融场景做 1: 1 的人脸识别，用于互联网金融的注册，这与在家用相册中做人脸识别，也就是把照片集中同一个人的照片找出来，以及在安防场景中，根据模糊的照片在一个海量的逃犯库中找到特定的人，这些场景都是人脸识别，准确率可能都差不多 99%、或者 99% 点几。虽然企业这么宣称，但实际背后蕴含的差异是非常大的，它会有非常多影响因素，所以准确率跟行业背景以及前置假设会是一个强相关的关系。而不同的场景下取得的识别准确率很难做类比。

相比不知前提的识别正确率，更为重要的是，在不同的场景下，企业是不是能够使用原创技术真正地取得突破。在互联网相册的应用场景下，商汤可以说是全世界第一个让计算机的人脸识别超越了人类，而后续很多智能相册的业务和服务都脱胎于这项突破。在杨帆看来，当公司面临一个新的行业场景，和过去的场景不一样且遇到新的挑战的时候，是不是能够

率先去形成量变的突破，这才是最重要的。当技术沉淀、数据积累和对业务场景的理解，三者融合在一起的时候，才能帮助公司完成一个真正有价值的、有意义的技术突破。

当识别率达到 99% 以后，人脸识别技术面临的难点主要在于，如何在不同行业场景中深化这项技术。虽然看上去 99% 的识别率已经很高了，但不同行业场景对于识别率的要求不同，99% 可能只是该技术得以使用的入门条件，比如银行身份认证服务，如今商汤人脸识别的误识别率已经可以做到 10^{-7} 次方，相当于 7 位银行密码，但在这个场景下也才刚刚得以使用；而安防场景下，照片模糊、有遮挡、角度不佳都给人脸识别带来了更现实的挑战。



“看似同质化很强、很简单的人脸识别，细分的技术场景其实非常复杂，所以脱离场景去谈技术是没有太大意义的，今天能看到的，包括以安防、手机这样的一些重点行业为代表，对于真正的人脸识别技术的全面深化存在着非常多的挑战，值得我们去攻克。”

图像和视频分析比你想像的更复杂

图像和视频分析其实是一个从功能或者从能力角度来看都比较复杂的

技术体系，当我们把一项技术落地或深化的时候，它可能需要几个团队合作完成。

商汤在计算机视觉技术领域的探索工作大致可以分为图像增强、物体检测和分类、算法模型、训练引擎等几个方面。

图像智能化增强是图像和视频分析的第一步，虽然今天照片和视频的采集设备已经非常好了，但图像和视频的采集还是经常面临困难，比如用红外摄像头以及结构光摄像头，拿到的深度图信息里面的噪音非常大，或者用安防设备拍摄高速运动的物体时会因为运动而导致模糊，因此分析前需要对这些图像和视频进行智能化的增强和恢复，又叫做 Low Level Vision，这在商汤是一项独立的工作，目的在于提升采集到的图像和视频的质量。

而图像和视频的识别及分析又可以细分成多个部分，包括物体检测，知道一个东西在哪里；物体的关键点定位，知道物体的关键轮廓和形状；物体的分类，就是对于找到的物体，能够知道它是什么东西；整个区域的分割，对整个物体的边缘或轮廓有非常清晰的描述。实际上，整个识别体系可能需要分成若干个不同的子领域，在真正的行业应用中，它往往是一些子领域叠加组合的应用。

商汤有专门的团队进行基础研究，比如如何将算法小型化，使之能够在资源受限的移动终端上运行；如何优化算法使之运行得更快；AI 核心的训练引擎或操作系统的持续升级和演进；弱监督或无监督学习的研究，包括增强学习、迁移学习等前沿技术。

杨帆强调，从计算引擎到数据流程架构，更重要的意义其实不在于数据量，而在于让算法形成一个稳定的闭环。

计算机视觉技术如何落地实际产品

计算机视觉技术在商汤的落地场景

商汤一直非常关注计算机视觉技术的落地，杨帆在早前的一些分享和

演讲中也多次提及技术进步需要与产业需求相结合。据杨帆介绍，计算机视觉技术在商汤的产品和业务中主要包含以下应用场景：

安防

过去对安防的理解主要是公安，其实真正意义上的安防还包括交通、线下的商业场景、小区、学校等，可以涵盖的场景非常大。

智能终端

目前智能终端主要指手机，但它未来的形态可能会继续演化，人工智能的技术一定会在这样的终端设备上体现出非常大的价值。

互联网视频类应用

随着互联网应用的进一步加深，它会越来越多地从文本转向图像、视频这种更加丰富的多媒体形态的应用，这些年从直播到短视频的爆发都是例子。在这方面，商汤可以给视频类应用的厂商提供非常完整而丰富的高附加值的解决方案。

人像身份认证

基于人像的身份认证也是一个非常有价值的工作，它是一个特殊的跨行业的解决方案。这个解决方案现在已经从线上到线下开始极大范围地蔓延。对中国来说，个人公民身份信息的实名制是一个非常重要的诉求，这个诉求能够有效地帮我们在一定程度上解决互联网的安全问题、解决线下的公共安全问题。所有线上的互联网行业应用，到各种线下行业，包括机场、超市、酒店，都会有越来越多的对于个人身份信息核验的强烈需求，商汤在这方面也提供了非常完整的解决方案。

自动驾驶

自动驾驶会是未来一个非常大的标杆性的方向，在这个过程中，人工智能技术会是一个非常关键的环节，商汤在这个领域也有一定的投入和规划。

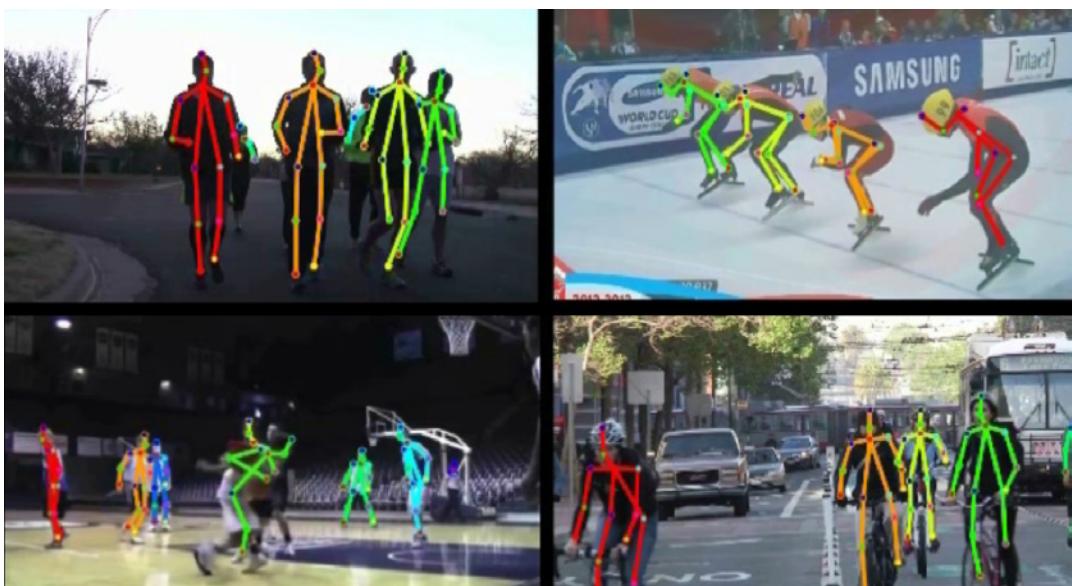
商汤安防场景背后的技术支撑

一款合格的安防产品，背后绝不只靠人脸识别这一项，而是由多项技

术共同支撑。

以一个广场级别的安防监控场景为例，其背后涉及的技术主要包括：

1. 硬件设备，即摄像头。对于大型广场，一个摄像头无法全面覆盖，因此可能需要全景摄像头和可拉伸的近景摄像头配合，完成人脸或其他图像的采集。
2. 采集算法。摄像头中会集成一个人群分析的算法，即通过收集的数据、结合人工规则，了解这个广场现在哪里人流比较密集、哪里人停留时间比较长，然后让负责抓拍和跟进的摄像头重点关注这些区域。
3. 人脸识别。接下来就可以在上述区域使用人脸识别的技术，寻找是否有黑名单（比如扒手库）中的人，可以用于反扒。这也是为什么刚才要找人密集的区域、停留时间长的区域，因为这些是高发区。
4. 肢体动作捕捉和识别。在寻找特定人员的过程中，需要进行人体姿态的跟踪，通过对这些人的关键动作进行检测和识别，从而判断是否出现偷窃行为。
5. 图像增强。如果摄像头采集到的图片模糊了，还会用到图像增强技术，使图像变得更适合后续步骤分析。



如杨帆所说，真正去看行业落地的时候，往往都是不同的技术叠加和组合的应用，这里面人脸识别和动作识别是最关键的技术，但实际上想把落地场景做好，一定需要多种技术组合。

复合型人才是 AI 落地的关键

杨帆表示，将创新技术转变为实际产品是一条满是荆棘的道路，行之不易，而其中最大的难点，一是如何选对方向和时机，二是如何找到合适的人才。

AI 技术落地需要与行业相结合，而如何去选择需要结合的行业就是第一个难题。杨帆说，“如果技术还没有到真正能成功的门槛，比如搜索引擎中的视频搜索，大公司不断积累可能没问题，但如果是一个小的创业公司，把它作为安身立命之本，难以得到回报，可能两年之后就死了。”杨帆表示，首先需要确认所选择的行业市场是一个真实有效、有规模的刚需市场；其次，需要在市场中真正拿到完整的闭环数据，才能获得持续性的进步；接下来，需要考虑行业当前的技术红线是不是在一个合理的区间内，介入太晚或介入太早，都是会有问题的；最后，在产品落地的过程中，需要考虑如何利用技术门槛期（通常 1 年到 1 年半）带来的优势，进一步建立行业壁垒，只有技术壁垒而没有行业壁垒的话，最后从长期来讲还是为他人做嫁衣。

从另一方面来讲，行业落地需要各种综合性的关键技术的整合。行业的需求往往是一些相对模糊的，而且从技术上来看是非常不明确的东西，这时候就需要有人有足够的能力去一一拆解。在杨帆看来，找到或培养一些既有技术背景、又对行业有足够深的理解的人才，是企业实现 AI 技术落地最关键的一点。他说到，“人才问题、团队组织问题、发展问题，特别是做 2B 行业，标准化与非标准之间的平衡性掌握，任何一个技术性产品落地会面临的共有问题，做 AI 技术落地，这些问题一个都不会少，而只会更严重。AI 人才是个更大的坑，AI 的技术性更深重，从过往来看，它跟行业的结合更弱，所以你想要真正去打磨出一个符合真正行业需求的

产品的时候，需要把对行业的理解和对技术的理解融合在一起，这在我看来是最有挑战的，因为过去可能这个世界上基本不存在这样的人，对行业有理解的人很少。”

市场增量期，商汤更愿意合作而非竞争

人工智能领域的创业浪潮中，计算机视觉技术（CV）在国内是一个非常火热的方向，呈遍地开花之势。在安防、金融、机器人、医疗、无人驾驶等诸多业务场景都有大批公司在竞争。

安防是商汤非常重要的一个业务场景，也是国内很多计算机视觉初创企业（如旷视科技、依图、云从等）非常看重的市场，更不用说已经在这个领域深耕多年的海康威视。

杨帆认为，安防市场目前正处于高速增长期，从 2018 到 2019 年，整个安防市场还会大爆发，爆发速度可能会超过大家的想象。而商汤的定位是依托原创技术去做能力服务平台，去做不同行业的赋能者，这使得商汤更愿意跟行业上下游企业形成合作而不是竞争的关系。

人脸识别技术的安全性问题

人脸识别技术多用于安防和金融领域，尤其像银行、支付相关的人脸识别应用对安全性要求特别高。前不久苹果发布会上推出的 FaceID 也引发了大家对于其是否足够安全的讨论。

杨帆将人脸识别的安全性问题分为两种，一种是人脸识别如何做得更准确，不会误识别；另一种则是如何防御非法攻击，比如通过照片、视频等方式绕开人脸识别。随着数据量的增大以及新算法的迭代演进，人脸识别的准确率一直在不断提升，相对而言，后一个问题面对的挑战更大，这个问题在业界又被称为活体检测问题。

对于金融场景的非法攻击防御，商汤目前的做法主要是通过积累大量的攻击数据，并通过模式分析、光谱分析等方法识别出攻击行为的模式，进而抵挡这些攻击。杨帆解释说：“不管用视频还是照片，其实有很多蛛

丝马迹是可以看到的，但这种蛛丝马迹人不一定能够特别好地分辨，当有大量数据的时候机器可以比较好地分辨，比如手机屏幕的反光等。”

苹果 FaceID 采用的 3D 人脸识别技术，主要的差异在于采集设备，将采集设备换成 3D 摄像头之后，能够采集到的图像数据信息更大，除了彩色信息之外，还会拥有 3D 的数据信息，而这些深度信息能够使算法进行更好的分析，从而达到更好的人脸识别以及防御攻击的效果。杨帆认为 3D 采集设备的研发和发展是一个比较明确的行业趋势，商汤未来在这个方向上也会做一些尝试。

计算机视觉技术的未来

对于计算机视觉技术目前面临的挑战，杨帆认为主要有三点，第一是如何减少对数据的依赖，而这也是行业内大家达成共识的一个大的方向，目前的图像识别模式对于数据依赖太强，人类识别的时候并不需要这么大量的数据。第二个是整体性能优化，就是如何用更低的计算成本完成智能分析，这对于实用化非常重要。第三个则是理论研究，知其所以然还是很重要的，这样更有助于长期发展。

杨帆认为视频的分析理解是未来计算机视觉比较有前景的研究方向之一。他说，“视频的分析理解，其实大家喊了很多年，到底什么时候算是真正成熟的点，不同的人会有不同的判断，会在不同的时期投入。我个人认为互联网作为一个已经成型的、具有特别大的商业价值的体系链，视频的应用在我看来是太少而不是太多。视频或者说视觉信号的潜在价值是非常大的，因为人和人之间沟通其实视觉信息占非常重要的比例，它的信息含量非常丰富。今天互联网已经形成了非常完整的生态，它对信息的五个环节都有特别好的基础技术支撑，在这种情况下，率先对视频领域做更深的探索和挖掘其实是必经之路。很多线下的行业可能有刚需，互联网上的视频、图像，特别是视频内容分析理解相关的领域，在未来其实还是会有很大的空间，今天能够做的事情还是太少。”

在整个人工智能布局上，计算机视觉的定位是怎样的？

视觉是最核心的，而且潜在商业价值也是最大的。

杨帆认为，信息是一切的核心，抛开人工智能，整个IT行业所做的事情就是信息的采集、传输、存储、分析、计算和反馈。而人工智能就是在整个信息环中，机器越来越多地去承担人的角色，可能比人做得更好。人和人日常进行交互的时候，视觉信息是更加本质的信息，所包含的信息量更大，因此计算机视觉在整个信息形态上是以一个相对高阶的形态存在，对各个环节的技术要求都会更高。一旦在每个环节上逐步具备视觉信息的处理能力之后，它所迸发出来的价值可能会超过今天IT互联网行业所能影响的空间，甚至可能会颠覆人和人、人和这个世界的交互。

在杨帆看来，计算机视觉有一个很重要的点，就是人的眼睛能够分析、感受的电磁波是一个很窄的波段，而机器却识别更宽的波段，比如红外摄像头、近红外摄像头、结构光深度的摄像头。杨帆提出了一个很有趣的问题：“这些摄像头能够把人类所能够看到的、能够处理的波段进一步扩展。那这个东西是不是可以一直扩展下去？如果从这个角度去理解，计算机视觉意味着将来机器可以替代人类，或者它作为人类的助手拥有更加本质的对这个世界的洞察。”

杨帆认为，目前我们设计、使用红外摄像头的方式思路还是从人出发的，依赖于人类经验的辅助和指导，也就是先将红外摄像头所采集到的影像信息，转化成一个人类可理解的影像，然后用机器去理解它。他说：“而下一步，很可能是红外摄像头直接去采集机器可以理解的信息形态，然后机器可以再去扩展。”



在微信上关注我们



InfoQ

国内最好的原创技术社区，一线互联网公司核心技术人员提供优质内容。订阅 InfoQ，看全球互联网技术最佳实践。做技术的不会没听过 QCon，不会不知道 InfoQ 吧？——冯大辉从事技术工作，或有兴趣了解 IT 技术行业的朋友，都值得订阅。——曹政



关注「InfoQ」回复“二叉树”，看十位大牛的技术初心，不同圈子程序员的众生相。



聊聊架构

以架构之“道”为基础，呈现更多的务实落地的架构内容。



关注「聊聊架构」
和百位架构师共聊架构



细说云计算

探讨云计算的一切，关注云平台架构、网络、存储与分发。这里有干货，也有闲聊。



关注「细说云计算」
回复“群分享”，
看云计算实践干货分享文章



AI前线

提供最新最全AI领域技术资讯、一线业界实践案例、业界技术分享干货、最新AI论文解读。



关注「AI前线」
回复“AI”，下载《AI前线》
系列迷你书



前端之巅

紧跟前端发展，共享一线技术，不断学习进步，攀登前端之巅。



关注「前端之巅」
回复“京东”，看京东
如何做网站前端监控



移动开发前线

关注移动开发领域最前沿和第一线开发技术，打造技术分享型社群。



关注「移动开发前线」
回复“群分享”，看移动
开发实践干货文章



高效开发运维

常规运维、亦或是崛起的DevOps，探讨如何IT交付实现价值。



关注「高效开发运维」
回复“DevOps”，四篇精品
文章领悟DevOps

QCon

全球软件开发大会2018

主办方 Geekbang 极客邦科技 InfoQ

[北京站]

北京·国际会议中心

演讲：2018年4月20–22日 培训：2018年4月18–19日

纵览20大热门专题

最低优惠7折进行时

现在报名每张立减2040元

移动支付及共享单车行业应用实践 人工智能与大数据处理技术 大前端实践

团购享受更多优惠 截至2017年12月31日

编程语言

Java前沿

前端前沿技术

大前端实践

新兴大数据处理技术

人工智能与业务实践

深度学习

运维新趋势

高可用架构

业务架构

微服务架构

大数据平台架构

大会官网：www.qconbeijing.com
访问官网获取更多前沿技术趋势

如有任何问题，欢迎咨询
电话：151 1001 9061
微信：qcon-0410





架构师 月刊 2017年11月

本期主要内容 : KRACK 可攻陷所有 WiFi 网络 ; Docker 官方将支持 Kubernetes ; 微博技术大 V 老师木的机器学习水平怎么样 ? 月活超美国人口十分之一 , 各大科技巨头纷纷布局 , 智能音箱背后有何门道 ? 做技术选型时 , 要注意些什么 ? 体系化认识 RPC 。



深度学习器： TensorFlow 程序设计

本书详细介绍了 TensorFlow 程序设计中的几个关键技术。



架构师双十一特刊： 电商大促技术探秘

今年我们尝试将技术内容转换为语音并推出了极客时间 APP , 思考和创新不会止步于此 , 希望来年双十一专题能以更棒的内容形式与大家再见面。



架构师特刊 范式大学

构建商业 AI 能力的五大要素 ; 判别 AI 改造企业的 70 个指标 ; 用最小成本 验证 AI 可行性 ; 企业技术人员如何向人工智能靠拢 ? 人工智能的下一个技术风口与商业风口。