36 | 当前技术的发展趋势以及其对编译技术的影响

2019-11-15 宫文学 来自北京

《编译原理之美》

在 IT 领域，技术一直在飞速的进步，而每次进步，都会带来新的业态和新的发展机遇。

退回到 10 年前，移动互联网刚兴起不久，谁也没想到它会催生现在这么多的业态。而云计算还在酝酿期，腾讯和百度的创始人都觉得它走不远，现在竟然这么普及。

退回到 20 年前，互联网刚兴起，上网都要拨号。互联网的几个巨头，像阿里巴巴、百度、腾讯、新浪，还有网易，都是在那个时代展露头角的。毫不夸张地说，如果你在那个时代搞技术，懂 Web 编程的话，那绝对是人人争抢的“香饽饽”，毕竟那时，Web 编程是前沿技术，懂这个领域的人，凤毛麟角。

退回到 30 年前，微软等公司才刚开始展露头角，雷军、求伯君等老一代程序员也正在发力，WPS 的第一个版本运行在 DOS 操作系统上。我还记得，95 年的时候，我在大学的阶梯教室里，看了比尔盖茨曾发表的，关于未来技术方向的演讲。当时，他预测了未来的科技成果，比

如移动智能设备，听上去像天方夜谭，但现在移动互联网、人工智能和 5G 的发展，早已超出了他当时的想象。

那么你有理由相信，未来 10 年、20 年、30 年，会发生同样天翻地覆的变化。这种变化所造成的的影响，你我哪怕大开“脑洞”都无法预料。而你在这种趋势下，所能做的就是，把握当下，并为未来的职业生涯做好准备。**这是一件认真且严肃的事情，值得你用心对待。**

当然，洞悉未来很难，但你可以根据当前了解到的信息，捕捉一些发展趋势，看看这些发展趋势，让编译技术的发展方向有了哪些不同，跟你又有什么关系。

本节课，我想与你分享 3 个方面的技术发展趋势，以及它们对编译技术的影响：

人工智能，以及如何让编程和编译技术变得更智能？

云计算，以及是否需要云原生的语言？

前端技术，以及能否出现统一各个平台的大前端技术？

期望这些内容，能让你看到一些不同的思考视角，获得一些新的信息。

趋势 1：让编程更智能

人工智能是当前发展最迅速的技术之一了。这几年，它的发展速度超过了人们的预期。那么你知道，它对编译技术和计算机语言的影响是什么吗？

首先，它需要编译器能够支撑，机器学习对庞大计算力的需求，同时兼容越来越多新的硬件架构。

由于机器学习的过程需要大量的计算，仅仅采用 CPU 效率很低，所以 GPU 和 TPU 等新的硬件架构得到了迅速的发展。对于编译技术来说，首要的任务，是要充分发挥这些新硬件的能力；因为 AI 的算法要能跑在各种后端架构上，包括 CPU、GPU 和 TPU，也包括仍然要采用 SIMD 等技术，所以后端技术就会变得比较复杂。同时，前端也有不同的技术框架，比如谷歌的 TensorFlow、Facebooke 的 PyTorch 等。那么编译器怎样更好地支持多种前端和多种后端呢？

根据在24 讲学到的知识，你应该会想到要借助中间代码。所以，MLIR 应运而生。**这里要注意，**ML 是 Multi-Level（多层次）的意思，而不是 Machine Learning 的缩写。**我还想告诉你，**MLIR 的作者，也是 LLVM 的核心作者、Swift 语言的发明人，Chris Lattner（他目前在谷歌 TensorFlow 项目中）。而当你看到 MLIR 的格式，也许会觉得跟 LLVM 的 IR 很像，那么你其实可以用更短的学习周期来掌握这个 IR。

其次，AI 还可能让现有的编译技术发生较大的改变。

实际上，把 AI 和编译技术更好地结合，是已经持续研究了 20 年左右的，一个研究方向。不过，没有很大的发展。因为之前，人工智能技术的进步不像这几年这么快。近几年，随着人工智能技术快速进步，在人脸识别、自动驾驶等各个领域产生了相当实用的成果，人们对人工智能可能给编译技术带来的改变，产生了更大的兴趣。这给了研究者们研究的动力，他们开始从各个角度探索变革的可能性。

比如说，在后端技术部分，很多算法都是 NP 完全的。这就是说，如果你用穷举的方法找出最优解，成本非常高。这个时候，就会用启发式（heuristic）的算法，也就是凭借直观或经验构造的算法，能够在可接受的花费下找出一个可行的解。那么采用人工智能技术，通过大数据的训练，有可能找出更好的启发式算法，供我们选择。这是人工智能和编译技术结合的一个方向。

Milepost GCC 项目早在 2009 年就发布了，它是一款开源的，人工智能编译器。它能够通过自动学习来确定去优化哪些代码，以便让程序的性能更高。据 IBM 的测试数据，某些嵌入式软件的性能因此提升了 18%。

另一个讨论度比较高的方向就是**人工智能编程（或者叫自动编程）。**从某种意义上看，从计算机诞生到现在，我们编写程序的方式一直没有太大的进步。最早，是通过在纸带或卡片上打孔，来写程序；后来产生了汇编语言和高级语言。但是，写程序的本质没有变化，我们只是在用更高级的方式打孔。

讽刺的是，在计算机语言的帮助下，很多领域都出现了非常好的工具，比如 CAD 完全改变了建筑设计行业。但计算机行业本身用的工具仍然是比较原始的，还是在一个编辑器中，用文本的方式输入代码。

而人工智能技术可能让我们习惯已久的编程模式发生改变。比如，现在的编译器只是检查错误并生成代码，带有 AI 功能的编译器呢，有可能不仅检查出比较明显的错误，甚至还会对你的编码方式提出建议。假设你用一个循环去做某个数组的计算，带有 AI 功能的编译器会告诉你，用函数式编程做向量计算性能更高，并提供一键式替换功能。

这里延伸一下，有可能，未来写程序的方式都会改变。微软收购 GitHub 以后，运用大量的代码作为训练数据，正在改进 IDE，提供智能提示功能。而这还只是开始。**目前，AI 其实已经能帮你做 UI 的设计：**你画一个草图，AI 给你生成对应的 Web 页面。

而且在 AI 辅助设计领域，算法还能根据你的需要，帮你生成平面或三维的设计方案。我能想象，未来，你告诉 AI 给你生成一个电商 APP，它就能给你生成出来。你再告诉它做什么样的修改，它都会立即修改。在未来，应用开发中，最令人头疼的需求变化的问题，在 AI 看来根本不是事。

那么，如果这个前景是真实的，**对于你而言，需要掌握什么技能呢?**

我建议你了解，编译技术和人工智能这两个领域的知识。那些计算机的基础知识会一直有用，你可以参与到编程范式迁移，这样一个伟大的进程中。现有程序开发中的那些简单枯燥，又不需要多少创造力的工作，也就是大家通常所说的“搬砖”工作，可能会被 AI 代替。而我猜测，未来的机会可能会留给两类人：

> 一类是具备更加深入的计算机基础技能，能应对未来挑战的，计算机技术人才，他们为新的计算基础设施的发展演化，贡献自己的力量。
>
> 另一类是应用领域的专家和人才。他们通过更富有创造力的工作，利用新的编程技术实现各种应用。编写应用程序的重点，可能不再是写代码，而是通过人工智能，训练出能够反映领域特点的模型。

当然，向自动编程转移的过程肯定是逐步实现的：AI 先是能帮一些小忙，解放我们一部分工作量，比如辅助做界面设计、智能提示；接着是能够自动生成某些小的、常用的模块；最后是能够生成和管理复杂的系统。

总而言之，AI 技术给编译技术，和编程模式带来了各种可能性，而你会见证这种转变。除此之外，云计算技术的普及和深化，也可能给编译技术和编程模式带来改变。

趋势 2：云原生的开发语言

云计算技术现在的普及度很广，所有应用的后端部分，缺省情况下都是跑在云平台上的，云就是应用的运行环境。

在课程里，我带你了解过程序的运行环境。那时，我们的关注点，还是在一个单机的环境上，包括 CPU 和内存这些硬件，以及操作系统这个软件，弄清楚程序跟它们互动的关系。比如，操作系统能够加载程序，能够帮程序管理内存，能够为程序提供一些系统功能（把数据写到磁盘上等等）。

然而，在云计算时代，**云就是应用的运行环境。**一个应用程序不是仅仅加载到一台物理机上，而是要能够根据需要，加载很多实例到很多机器上，实现横向扩展。当然了，云也给应用程序提供各种系统功能，比如云存储功能，它就像一台单独的服务器，会给程序提供读写磁盘的能力一样。

除此之外，在单机环境下，传统的应用程序，是通过函数或方法，来调用另一个模块的功能，函数调用的层次体现为栈里一个个栈桢的叠加，编译器要能够形成正确的栈桢，实现自动的内存管理。**而在云环境下，**软件模块以服务的形式存在，也就是说，一个模块通过 RESTful 接口或各种 RPC 协议，调用另外的模块的功能。程序还需要处理通讯失败的情况，甚至要在调用多个微服务时，保证分布式事务特性。而我们却没从编译技术的角度，帮助程序员减轻这个负担。

导致的结果是：现在后端的程序特别复杂。你要写很多代码，来处理 RPC、消息、分布式事务、数据库分片等逻辑，还要跟各种库、框架、通讯协议等等打交道。**更糟糕的是，**这些技术性的逻辑跟应用逻辑，混杂在一起，让系统的复杂度迅速提高，开发成本迅速提升，维护难度也增加。很多试图采用微服务架构的项目因此受到挫折，甚至回到单一应用的架构。

所以，一个有意义的问题是：能否在语言设计的时候，就充分利用云的基础设施，实现云原生（Cloud Native）的应用？也就是说，我们的应用，能够透明地利用好云计算的能力，并能

兼容各种不同厂商的云计算平台，就像传统应用程序，能够编译成，不同操作系统的可执行文件一样。

好消息是，云计算的服务商在不断地升级技术，希望能帮助应用程序，更好地利用云计算的能力。而无服务器（Serverless）架构就是最新的成果之一。采用无服务器架构，你的程序都不需要知道容器的存在，也不需要关心虚拟机和物理机器，你只需要写一个个的函数，来完成功能就可以了。至于这个函数所需要的计算能力、存储能力，想要多少就有多少。

但是，云计算厂商提供的服务和接口缺少标准化，当你把大量应用都部署到某个云平台的时候，可能会被厂商锁定。如果有一门面向云原生应用的编程语言，和相应的开发平台，能帮助人们简化云应用的开发，同时又具备跨不同云平台的能力，**那就最理想了。**

实际上，已经有几个创业项目在做这个方向做探索了，比如 Ballerina、 Pulumi和 Dark，你可以看一下。

当然了，云计算和编程结合起来，就是另一个有趣的话题：云编程。我会在下一讲，与你进一步讨论这个话题。

趋势 3：大前端技术栈

上面所讲的云计算，针对的是后端编程，而与此对应的，是前端编程工作。

后端工作的特点，是越来越云化，让工程师们头疼的问题，是处理分布式计算环境下，软件系统的复杂性。**当然，前端的挑战也不少。**

我们开发一款应用，通常需要支持 Web、IOS 和 Android 三种平台，有时候，甚至需要提供 Windows 和 macOS 的桌面客户端。不同的平台会需要不同的技术栈，从而导致一款应用的开发成本很高，这也是前端工程师们不太满意的地方。

所以，前端工程师们一直希望能用一套技术栈，搞定多个平台。比如，尝试用 Web 开发的技术栈完成 Android、IOS 和桌面应用的开发。React Native、Electron 等框架是这个方面的有益探索；Flutter 项目也做了一些更大胆的尝试。

Flutter 采用 Dart 开发语言，可以在 Android 和 IOS 上生成高质量的原生界面，甚至还可以支持 macOS、Windows 和 Linux 上的原生界面。另外，它还能编译成 Web 应用。所以，本质上，你可以用同一套代码，给移动端、桌面端和 Web 端开发 UI。

你可以把这种技术思路叫做大前端：同一套代码，支持多个平台。

从 Flutter 框架中，你可以看出编译技术起到的作用。首先，Dart 语言也是基于虚拟机的，编译方式支持 AOT 和 JIT，能够运行在移动端和桌面端，能够调用本地操作系统的功能。对于 Web 应用则编译成 JavaScript、CSS 和 HTML。这个项目的创新力度已经超过了 React Native 这些项目，工程师们已经不满足于，在现有的语言（JavaScript）基础上编写框架，而是用一门新的语言去整合多个技术栈。

当然，提到前端技术，就不能不提 Web Assembly（WASM）。WASM 是一种二进制的字节码，也就是一种新的 IR，能够在浏览器里运行。相比 JavaScript，它有以下特点：

静态类型；

性能更高；

支持 C/C++/Rust 等各种语言生成 WASM，LLVM 也给了 WASM 很好的支持；

字节码尺寸比较少，减少了下载时间；

因为提前编译成字节码，因此相比 JavaScript 减少了代码解析的时间。

由于这些特点，WASM 可以在浏览器里，更高效地运行，比如可以支持更复杂的游戏效果。**我猜想，**未来可能出现，基于浏览器的、性能堪比本地应用的字处理软件、电子表格软件。基于云的文档软件（比如 Google Doc）会得到再一次升级，使用者也将获得更好的体验。

此外，WASM 还允许除了 JavaScript 之外的语言，来编写 Web 应用。这些语言可以像 JVM 上的语言一样，生成字节码，并且只要有运行 WASM 的虚拟机，它们就具备一样的可移植性。

而且，WASM 不仅可以运行在前端，还可以运行在后端。就像 JavaScript 语言被 Node.js 项目，用于开发后端服务一样，现在 Node.js 项目也可以调用 WASM 模块。还有一些更激进的

项目，正在开发高效运行 WASM 的虚拟机，比如wasmer 项目。wasmer 虚拟机可以使用 LLVM 进行编译和优化，从而能够提供更高的性能。

讨论到这里，你有什么感受？C/C++ 语言写的程序，以 WASM 的形式运行在浏览器里，或者运行在后端的虚拟机里，通过即时编译运行。完全颠覆了你对这两门语言的传统印象吧？这就是编译技术与时俱进的一个体现。

其实，学过《编译原理之美》这门课程以后，我也期望你有信心，做一款 WASM 的虚拟机，并基于它，做一个类似 Node.js 的后端服务平台。因为这并没有太大的技术难度，你只要做到稳定好用，花费很多心血就是了。

课程小结

为了拓展你的视野，我带你探讨了三个技术的发展趋势，以及它们对编译技术和编程方式所带来的影响。我希望，在学完本节课之后，你能有以下收获：

- 人工智能有可能提升现有的编译技术框架，并带来自动编程等，编程模式的重大变化。
- 应用程序的运行环境，不能仅仅考虑单机，还要考虑云这个更大的环境。因此，新一代的编程语言和开发平台，可能会让开发云原生的应用更加简单。
- 在应用开发的前端技术方面，如果要想支持多种平台，可能还需要通过编译技术来获得大的突破。

当然，编译技术还有很多其他的研究方向，比如更好地支持并行计算、支持物联网和低功耗场景，支持区块链，甚至支持一些同学感兴趣的，未来的量子计算机，等等。**不过，在我看来，**我在文中提到的这三个趋势，跟你的关系是最为密切的。因为你现在或多或少地都在接触 AI、云和前端技术。

我希望今天的内容能帮你开拓思路，为迎接未来的技术趋势做好准备，并且能够更好地利用编译技术，增强自身的竞争力。

一课一思

在本节课中，我分享了自己对技术趋势的思考和感悟，而你或许有其他的见解，欢迎在留言区与我讨论，碰撞思维的火花。

感谢你的阅读，如果这篇文章让你有所收获，也欢迎你将它分享给更多的朋友。

精选留言 (6)

Milittle
2019-11-16

这些我的大脑有曾想过，我认为在未来的世界中，我们不需要带手机 可能只需要带一块镶嵌在肌肤中的一块芯片就可以实现好多需求，比如全息影像就可以投影出我们想看的所有东西。然后通过通信技术的不断提高，我们万物互联。随时可以对远程的东西进行操控，游戏行业也会出现类似头号玩家中的场景。这样中国乃至世界都是互联的。当然这个过程是曲折的，但我相信所有的技术都会在这场革命中产生它的作用。（区块链 完全不需要出门带cash，中国前一段时间发布了虚拟货币，我把这个叫做free-cash）。

> 作者回复: 推荐我的朋友写的一套科幻小说：《云球》。
> 此人是中间件领域某公司的高管，放着几百万薪水的高管不当，潜心写小说去了... 以他雄厚的IT背景，写的科幻小说别具风格。
> 话说，IT背景的兄弟应该文理结合，去文学领域打劫一下也无不可...

共 3 条评论 >　　13

拉欧
2019-12-12

云原生这个大趋势是能感受的到的，因为是纯后端，大前端这个就不太了解了

> 作者回复: 前端平台呈现越来越多元化的态势。未来开发程序，可能还要支持鸿蒙等新的平台。
> Flutter这个项目虽然还不怎么成熟，却已经获得了极大的关注，就表现出程序员们想要用同一个技术栈，为各个前端平台开发应用的需求。

共 2 条评论 >　　2

- - -

xianhai
2019-11-15

脑洞大开

作者回复: :)

2

陌兮
2022-06-10

看老师的课程，真的是一种享受。但是我又开始焦虑了T^T。我的基础太差了

Geek_656245
2021-10-16

5g时代才真正开启前端时代，现在的下载资源已经没有过去那么廉价了，很多东西不在需要一个独立的客户端了，以后会把更多的业务放在前端来实现，免去下载这个功能了。

G二小店
2020-05-20

老师人工智能那一块 让我想起了陈天奇的tvm

作者回复: 人工智能程序，特别需要能够充分利用硬件的能力，因为所需要的计算量太大了。所以，AI框架需要深度使用编译技术。