开篇词-在真实世界的编译器中游历

你好，我是宫文学，一名技术创业者，现在是北京物演科技的CEO，很高兴在这里跟你见面。

我在IT领域里已经工作有20多年了。这其中，我个人比较感兴趣的，也是倾注时间精力最多的，是做基础平台类的软件，比如国内最早一批的BPM平台、BI平台，以及低代码/无代码开发平台（那时还没有这个名字）等。这些软件之所以会被称为平台，很重要的原因就是拥有很强的定制能力，比如流程定制、界面定制、业务逻辑定制，等等。而这些定制能力，依托的就是编译技术。

在前几年，我参与了一些与微服务有关的项目。我发现，前些年大家普遍关注那些技术问题，比如有状态的服务（Stateful Service）的横向扩展问题，在云原生、Serverless、FaaS等新技术满天飞的时代，不但没能被很好地解决，反而更恶化了。究其原因就是，状态管理还是被简单地交给数据库，而云计算的场景使得数据库的压力更大了，数据库原来在性能和扩展能力上的短板，就更加显著了。

而比较好的解决思路之一，就是大胆采用新的计算范式，发明新的计算机语言，所以我也有意想自己动手搞一下。

我从去年开始做设计，已经鼓捣了一阵了，采用了一些很前卫的理念，比如云原生的并发调度、基于Actor的数据管理等。总的目标，是要让开发云原生的、有状态的应用，像开发一个简单的单机应用一样容易。那我们就最好能把云架构和状态管理的细节给抽象掉，从而极大地降低成本、减少错误。而为编程提供更高的抽象层次，从来就是编译技术的职责。

Serverless和FaaS已经把无状态服务的架构细节透明掉了。但针对有状态的服务，目前还没有答案。对我而言，这是个有趣的课题。
在我比较熟悉的企业应用领域，ERP的鼻祖SAP、SaaS的鼻祖SalesForce，都用自己的语言开发应用，很可惜国内的企业软件厂商还没有做到这一点。而在云计算时代，设计这样一门语言绕不过去的一个问题，就是解决有状态服务的云化问题。我希望能为解决这个问题提供一个新工具。当然，这个工具必须是开源的。

正是因为给自己挖了这么大一个坑，也促使我更关心编译技术的各种前沿动态，也非常想把这些前沿的动态、理念，以及自己的一些实战经验都分享出来。

所以去年呢，我在极客时间上开了一门课程《编译原理之美》，帮你系统梳理了编译技术最核心的概念、理论和算法。不过在做第一季的过程中呢，我发现很多同学都跟我反馈：我确实理解了编译技术的相关原理、概念、算法等，但是有没有更直接的方式，能让我更加深入地把知识与实践相结合呢？

为什么要解析真实编译器？

说到把编译技术的知识与实践相结合，无外乎就是解决以下问题：

- 我已经知道，语法分析有自顶向下的方法和自底向上的方法，但要自己动手实现的话，到底该选择哪个方法呢？是应该自己手写，还是用工具生成呢？
- 我已经知道，在语义分析的过程中要做引用消解、类型检查，并且会用到符号表。那具体到自己熟悉的语言，这些工作是如何完成的呢？有什么难点和实现技巧呢？符号表又被设计成什么样子呢？
- 我已经知道，编译器中会使用IR，但实际使用中的IR到底是什么样子的呢？使用什么数据结构呢？完成不同的处理任务，是否需要不同的IR呢？
- 我已经知道，编译器要做很多优化工作，但针对自己熟悉的语言，这些优化是如何发生的？哪些优化最重

要？又要如何写出便于编译器优化的代码呢？

类似的问题还有很多，但总结起来其实就是：**真实世界的编译器，到底是怎么写出来的？**

那弄明白了这个问题，到底对我们有什么帮助呢？

第一，研究这些语言的编译机制，能直接提高我们的技术水平。

一方面，深入了解自己使用的语言的编译器，会有助于你吃透这门语言的核心特性，更好地运用它，从而让自己向着专家级别的工程师进军。举个例子，国内某互联网公司的员工，就曾经向Oracle公司提交了HotSpot的高质量补丁，因为他们在工作中发现了JVM编译器的一些不足。那么，你是不是也有可能把一门语言吃得这么透呢？

另一方面，IT技术的进化速度是很快的，作为技术人，我们需要迅速跟上技术更迭的速度。而这些现代语言的编译器，往往就是整合了最前沿的技术。比如，Java的JIT编译器和JavaScript的V8编译器，它们都不约而同地采用了“Sea of Nodes”的IR来做优化，这是为什么呢？这种IR有什么优势呢？这些问题我们都需要迅速弄清楚。

第二，阅读语言编译器的源码，是高效学习编译原理的重要路径。

传统上，我们学习编译原理，总是要先学一大堆的理论和算法，理解起来非常困难，让人望而生畏。

这个方法本身没有错，因为我们学习任何知识，都要掌握其中的原理。不过，这样可能离实现一款实用的编译器还有相当的距离。

那么根据我的经验，学习编译原理的一个有效途径，就是阅读真实世界中编译器的源代码，跟踪它的执行过程，弄懂它的运行机制。因为只要你会写程序，就能读懂代码。既然能读懂代码，那为什么不直接去阅读编译器的源代码呢？在开源的时代，源代码就是一个巨大的知识宝库。面对这个宝库，我们为什么不进去尽情搜刮呢？想带走多少就带走多少，没人拦着。

当然，你可能会犯嘀咕：**编译器的代码一般都比较难吧？以我的水平，能看懂吗？**

是会有这个问题。当我们面对一大堆代码的时候，很容易迷路，抓不住其中的重点和核心逻辑。不过没关系，有我呢。在本课程中，我会给你带路，并把地图准备好，带你走完这次探险之旅。而当你确实把握了编译器的脉络以后，你对自己的技术自信心会提升一大截。这些计算机语言，就被你摘掉了神秘的面纱。

俗话说“读万卷书，行万里路”。如果说了解编译原理的基础理论和算法是读书的过程，那么探索真实世界里的编译器是什么样子，就是行路的过程了。根据我的体会，**当你真正了解了身边的语言的编译器是怎样编写的之后，那些抽象的理论就会变得生动和具体，你也就会在编译技术领域里往前跨出一大步了。**

我们可以解析哪些语言的编译器？

那你可能要问了，在本课程中，**我都选择了哪些语言的编译器呢？选择这些编译器的原因又是什么呢？**

这次，我要带你解析的编译器还真不少，包括了Java编译器（javac）、Java的JIT编译器（Graal）、Python编译器（CPython）、JavaScript编译器（V8）、Julia语言的编译器、Go语言的编译器（gc），以及MySQL的编译器，并且在讲并行的时候，还涉及了Erlang的编译器。

我选择剖析这些语言的编译器，有三方面的原因：

- 第一，它们足够有代表性，是你在平时很可能会用到的。这些语言中，除了Julia比较小众外，都比较流行。而且，虽然Julia没那么有名，但它使用的LLVM工具很重要。因为LLVM为Swift、Rust、C++、C等多种语言提供了优化和后端的支持，所以Julia也不缺乏代表性。
- 第二，它们采用了各种不同的编译技术。这些编译器，有的是编译静态类型的语言，有的是动态类型的语言；有的是即时编译（JIT），有的是提前编译（AOT）；有高级语言，也有DSL（SQL）；解释执行的话，有的是用栈机（Stack Machine），有的是用寄存器机，等等。不同的语言特性，就导致了编译器采用的技术会存在各种差异，从而更加有利于你开阔视野。
- 第三，通过研究多种编译器，你可以多次迭代对编译器的认知过程，并通过分析对比，发现这些编译器之间的异同点，探究其中的原因，激发出更多的思考，从而得到更全面的、更深入的认知。

看到这里，你可能会有所疑虑：**有些语言我没用过，不怎么了解，怎么办？**其实没关系。因为现代的高级语言，其实相似度很高。

一方面，对于不熟悉的语言，虽然你不能熟练地用它们来做项目，但是写一些基本的、试验性的程序，研究它的实现机制，是没有什么问题的。

另一方面，学习编译原理的人会练就一项基本功，那就是更容易掌握一门语言的本质。特别是我这一季的课程，就是要帮你成为钻到了铁扇公主肚子里的孙悟空。研究某一种语言的编译器，当然有助于你通过“捷径”去深入地理解它。

我是如何规划课程模块的？

这门课程的目标，是要让你对现代语言的编译器的结构、所采用的算法以及设计上的权衡，都获得比较真切的认识。其最终结果是，如果要你使用编译技术来完成一个项目，你会心里非常有数，知道应该在什么地方使用什么技术。因为你不仅懂得原理，更有很多实际编译器的设计和实现的思路作为你的决策依据。

为了达到本课程的目标，我仔细规划了课程的内容，将其划分为预备知识篇、真实编译器解析篇和现代语言设计篇三部分。

在**预备知识篇**，我会简明扼要地帮你重温一下编译原理的知识体系，让你对这些关键概念的理解变得更清晰。磨刀不误砍柴工，你学完预备知识篇后，再去看各种语言编译器的源代码和相关文档时，至少不会被各种名词、术语搞晕，也能更好地建立具体实现跟原理之间的关联，能互相印证它们。

在**真实编译器解析篇**，我会带你研究语言编译器的源代码，跟踪它们的运行过程，分析编译过程的每一步是如何实现的，并对有特点的编译技术点加以分析和点评。这样，我们在研究了Java、Java JIT、Python、JavaScript、Julia、Go、MySQL这7个编译器以后，就相当于把编译原理印证了7遍。

在**现代语言设计篇**，我会带你分析和总结前面已经研究过的编译器，进一步提升你对相关编译技术的认知高度。学完这一模块以后，你对于如何设计编译器的前端、中端、后端、运行时，都会有比较全面的了解，知道如何在不同的技术路线之间做取舍。

好了，以上就是这一季课程的模块划分思路了。你会发现，这次的课程设计，除了以研究真实编译器为主要手段外，会更加致力于扩大你的知识版图、增加你的见识，达到“行万里路”的目的。

可以说，我在设计和组织这一季课程时，花了大量的时间准备。因此这一季课程的内容，不说是独一无二的，也差不多了。你在市面上很少能找到解析实际编译器的书籍和资料，这里面的很多内容，都是在我自己阅读源代码、跟踪源代码执行过程的基础上梳理出来的。

写在最后

近些年，编译技术在全球范围内的进步速度很快。比如，你在学习Graal编译器的时候，你可以先去看看，市面上有多少篇围绕它的高质量论文。所以呢，作为老师，我觉得我有责任引导你去看到、理解并抓住这些技术前沿。

我也有一个感觉，在未来10年左右，中国在编译技术领域，也会逐步有拿得出手的作品出来，甚至会有我们独特的创新之处，就像我们当前在互联网、5G等领域中做到的一样。

虽然这个课程不可能涵盖编译技术领域所有的创新点，但我相信，你在其中投入的时间和精力是值得的。你通过我课程中教给你的方法，可以对你所使用的语言产生更加深入的认知，对编译器的内部结构和原理有清晰理解。最重要的是，对于如何采用编译技术来解决实际问题，你也会有能力做出正确的决策。

这样，这个课程就能起到抛砖引玉的作用，让我们能够成为大胆探索、勇于创新的群体的一份子。未来中国在编译技术的进步，就很可能有来自我们的贡献。我们一起加油！

最后，我还想正式认识一下你。你可以在留言区里做个自我介绍，和我聊聊，你目前学习编译原理的最大难点在哪？或者，你也可以聊聊你对编译原理都有哪些独特的思考和体验，欢迎在留言区和我交流讨论。

好了，让我们正式开始编译之旅吧！

精选留言：

- sugar 2020-06-01 17:10:35
  终于见到宫老师的第二季啦～ 我来抢个首赞 哈哈 [4赞]

- 罗洪涛@融众 2020-06-01 18:15:58
  关注前端AST希望落地前端定制化 [1赞]

- X！！ 2020-06-01 17:08:06
  第一 [1赞]

- 贾献华 2020-06-01 23:52:11
  我要看懂汇编

- 至今未来 2020-06-01 23:51:15
  编译原理之美 只看了一遍 差不多忘光了 宫老师 我又来了(「･ω･)「嘿

- JoKERsunLay 2020-06-01 23:16:58
  老师您好！ 刚刚啃完Stanford的Compiler课 跟着写了lexer到cgen四个部分还是挺有意思的 很多知识也和老师的编译原理之美能相互印证

  想过来第二季实战一下

  另外请问老师会考虑加餐ML compiler的内容吗？

- humor 2020-06-01 21:40:19
  上一季的算法篇理解的不是很深入，感觉算法部分偏难。

- 吃鱼 2020-06-01 21:14:07
  老师，因为专业要学习二进制安全，所以特别想通过您的课程了解编译方面的知识，我编译原理之前学的不太扎实，您的两个课程我觉得都很硬核，应该先学哪一门比较好呢？

- 小晏子 2020-06-01 20:57:10
  最期待能把学的知识用在实际工作中

- Matrix 2020-06-01 18:55:20
  目前是在校研究生，研究方向是二进制的漏洞挖掘与利用。平常在论文、工程实现中多多少少和编译原理相关的知识有交集，如：SSA、AST、LLVM-IR等，相关的理论知识书本上学过，但没有形成较为清晰的知识体系，很多地方有一种雾里看花的感觉，希望能结合实际对编译器的内部结构和原理有更清晰的理解。

- 王成⊠ 2020-06-01 18:24:46
  老师好
  学习编译的难点:编译原理之美还没有学完，正在努力学习，由于工作学习等多方面原因，学习进度较慢
  打算应用编译原理实现的东西:目前工作是实时计算，公司目前关于实时流使用了storm和flink,我想开发一套程序，使得一次开发，可以同时在两个平台运行，同时，可以做到将一个平台的代码迅速转为另一个平台可以运行的代码

- Fan 2020-06-01 18:01:49
  期待宫老师多讲讲llvm方面相关的。