编程语言 Java Java 虚拟机（JVM） Clojure

关注者 465 被浏览 55,160

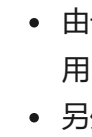
为什么很多语言选择在JVM上实现？

如题，最初看到有Jython和JRuby倒也觉得没什么，毕竟用户不算多。但是有些比较新的语言比如Scala、比如Clojure都选择在JVM上实现，这让我有些不解，为什么它们选择JVM？仅仅是为了跨平台吗？除了跨平台的特点，JVM还有什么其它的优势？

关注问题 写回答 2 条评论 分享 邀请回答 ··· 收起

21 个回答

默认排序



张西家
自由职业

74 人赞同了该回答

- 非常经济地实现跨平台。你的语言编译器后端只需要输出 JVM 字节码就可以。跨平台需要极大的工作量，举个例子，只是独立开发生成本地代码，就需要花费大量精力去针对不同平台和处理器进行优化（比如 Firefox 就会考虑重用 WebKit 的这部分后端）。
- JVM 卓越的 JIT （Just-In-Time 即时编译）性能。 JIT 可以在运行中记录程序运行的特征，并在其基础上做大量的优化（Java 企业级应用的优秀性能很大程度上是由此而来）。 JIT 自从 HotSpot JVM 随 Java 1.2 发布以来，JVM JIT 的性能不断提高，是无可争议的成功产品。把 JVM 作为目标平台意味着大量的性能优化工作可以「外包」给 JVM 来做，大大缩减了 Guest 语言的开发预算。
- 已经有多个成熟的实例，有大量的经验可以借鉴
- JVM 作为一个成熟的高层运行环境，为 Guest 语言提供了很多运行时所需要的服务，比如内存管理（有业界领先的垃圾回收等），很大程度上避免了额外的独立开发。
- JVM 有多个独立实现，也有若干厂商会持续推进，资料完备，社区巨大。
- Java 社区有大量成熟的库，一般来说，运行在 JVM 上的其它语言都会设计一个专用的「桥」来帮助直接使用 Java 的库，对潜在客户来说是个很好的卖点。
- Java 有还算不错的开发工具和环境。目标为 JVM 的很多语言会考虑用 Java 语言实现（至少在 bootstrap 阶段）。

题外话：

- 由于 Scala 的独特优势，现在有不少研究性语言会用 Scala 实现，可以看到这里会有间接地使用 JVM 。
- 另外一个趋势是把 Javascript 成为新的目标平台。很多主流语言都已经出现了编译器可以翻译成 Javascript[1]，这也是得益于近年来 Javascript 虚拟机性能的显著提升。

[1]: github.com/jashkenas/co...

编辑于 2012-01-09

▲ 74 ▼ 4 条评论 分享 收藏 感谢

RednaxelaFX
编程、编译原理、编程语言 等 7 个话题的优秀回答者

54 人赞同了该回答

放个传送门，我在SDCC 2012上做的演讲：莫枢：JVM是多语言的平台
<- 上面是CSDN的链接，不过显然他们在渲染我的演示稿的时候出了点格式问题。可以上SlideShare的同学可以看我自己上传的版本：JVM: A Platform for Multiple Languages

发布于 2015-08-19

▲ 54 ▼ 5 条评论 分享 收藏 感谢

Belleve
编程 话题的优秀回答者

21 人赞同了该回答

一个完整的语言有啥呢
前端、优化、后端、runtime、库
JVM 把后面四个都给包办了

ps. 对于 llvm，他包办的是 2 和 3，不过如果你语言是 target to native，4 不存在，5 本身不太困难

相关问题

为什么java会有jvm，其它语言怎么没有对应的vm？ 13 个回答

自创一门语言 是否可以利用JVM? 19 个回答

有必要使用Java以外的JVM语言吗？ 11 个回答

如何实现一门基于 JVM 的程序语言？ 3 个回答

如何评价 Clojure 语言的设计？ 13 个回答

相关推荐

编程小白学 Python
kula 等
240,406 人读过 阅读

编辑于 2016-04-19

▲ 21 ▼ 6 条评论 分享 收藏 感谢

徐明明
软件攻城师傅

6 人赞同了该回答

因为JVM有这个世界上最大的语言生态系统，基于它实现很多东西直接就有了，比如大量的三方库，成熟的JIT等等；又因为是基于JVM的，愿意尝试的人群会比较多，相对来说流行、成功的可能性会大一些。

发布于 2013-01-26

▲ 6 ▼ 添加评论 分享 收藏 感谢

匿名用户

10 人赞同了该回答

事实上，采用JVM实现的优点在于编译器的开发更容易。

编译器是分前端和后端的，参见前端和后端

在编译器中，**前端**将程序设计语言源代码转换成一种中间形式，然后**后端**再将它转换成计算机能够运行的二进制代码。**后端**还经常对代码进行优化以提升程序的运行效率。**前端**和**后端**的区别能够将处理原代码的语法解析器和生成机器码和对代码进行优化的**后端**去分开。一些**编译器**，像GCC，提供不同的**前端**分别解析不同语言的源代码 和/或 不同的**后端**针对不同的目标机器生成机器码。

显然，针对不同架构(ARM x86 等等)的CPU都为自己的语言编写一个新的后端是一件费时费力的事情，一门新的语言往往不能像c/c++那样有几十年锲而不舍的完善与补充。

快速地将一门语言开发出来推向实际应用才是最重要的。因此，后端直接扔给jvm；前端只需要做好语法规定，采用lex 和yacc等现成工具进行开发，很快就能做出来一个新语言的编译器。

快速原型，快速迭代，这才是现代开发的要求。

编辑于 2015-08-19

▲ 10 ▼ 2 条评论 分享 收藏 感谢

SEVEN
莫问收获，但问耕耘

4 人赞同了该回答

企业在选择一项技术的时候更多的考虑是技术本身带来什么，即实用性。而不是技术是否华丽，优美

基于上面的原则JVM的优势很明显：

1.成熟，标准，这一套实现已经沿用了很多年，得到过充分的验证，它代表了一个成熟的流派，包括后来的.Net FrameWork也是沿用同样的原理。

2.垃圾回收 GC这个概念在Lisp时代就已经定义，可能很多追求性能的同学会鄙视之，但无论如何它确实带来了开发效率的提升。

3.内存管理 其实这一点和2有些重复，但JVM的内存模型，基于栈的方式，方法执行的优化，甚至线程方面的处理都是非常经典和值得借鉴的

4.相关文档非常完善，标准清晰，有专门的公司推进，拥有大量的开发者。

但是有了上面几点并不一定代表你要用它，因为现实中的开发一定是要根据实际情况做出抉择的，而不是听信“某某技术牛X”这样的言论。

如果楼主有兴趣可以研究下JVM的实现，包括现在Android的Dalvik,甚至可以去Mono社区看看，里面很多东西还是很有意思的^_^

编辑于 2012-01-04

▲ 4 ▼ 添加评论 分享 收藏 感谢

ylxfc

3 人赞同了该回答

就目前而言，jvm是所有vm中做的最好的

发布于 2016-02-23

▲ 3 ▼ 添加评论 分享 收藏 感谢

圆胖肿
程序猿

4 人赞同了该回答

假设你有一个新的语言，你的目标是让越来越多人学会并使用这门语言

那你需要什么工具来帮你完成这个目标呢？除了推广你的语言语法以外，你还需要以下几个东西：

1. 跨平台，平台越多受众越广，除非性能下降到影响客户体验的程度，否则跨平台是最理想的
2. 你要有一个开发工具，ide最好，没有ide，至少也要有一个editor吧，好歹高亮一下，让别人写的时候更酥服
3. 你需要有一个包管理器，而且最好有一个make工具类似于c的make，能够直接完成编译打包这些事，然后包打出来了要有地方分享啊
4. 你需要有一个框架一样的东西，能够简化开发流程

那怎么做呢？以一个新语言要推广为例

首先第一个，跨平台，jvm最理想了，对吧？所以作者第一步需要把语言搬上jvm，如果你的语言是脚本的话，这一步会很简单，一个jsr223就可以了，但是如果是编译型的，会略麻烦一点你需要自己去捣鼓bytecode，但是都不是不可行，都可以做到，看你自己的功力

第二步，编辑器开发环境，这时候你可以在idea或者eclipse上捣鼓出来一个plugins，这个也可以做到，实际上很多语言都有自己的idea/eclipse plugin，这样你就不需要自己去搞一个ide啦，开发一个plugin就行啦

第三步，包管理工具，你需要有一个make工具，最理想的当然是maven central啦，支持maven, gradle, sbt等一大堆工具，你选一个自己喜欢的，然后在这个工具上开发plugin，这样你很快就有一个自己的make啦，这样你就不需要自己从头到尾做一个make啦，开发一个plugin就搞定啦，连包的repo这些东西，maven central这些都给你解决啦，要不然你自己还要去找一个repo，管理下载这些，很麻烦不是？

第四步，框架，找vert.x，vert.x是至今为止，对新语言最为友好的框架，没有之一，这样你的新语言就可以很快被用于生产并被用户尝试使用，也是目前polyglot执行最为彻底的框架，没有之一，它的源代码全部公开，codegen等内部api你学会了之后，就可以包装出来了，只要你能在vert.x上支持该语言，用户就可以很方便地尝试使用你的语言，如果连尝试都很麻烦的话，这个东西是没有前途的，因为要换太多东西，成本太高的话，很多人就放弃尝试了，所以polyglot可以降低用户尝试的成本

1，2，3，4，都没那么简单，也没有那么不可行，做都是可以做到的，但是工作量都不小

综合起来看，还是jvm+idea+maven+vert.x最方便了，全部都实现的话，基本上就可以在生产中尝试了，你需要的跨平台，编辑器，包管理工具，框架，完整了

我们使用新语言，也基本上是会看这1234都满足了没有

如果没有满足的话，我们暂不考虑使用，除非像是haskell,lisp这种已经被证明过是很牛逼的语言，那愿意贡献一下，让他变得更加贴近我们的需求，但是你自己发明的语言，什么都没有，你要别人参与并贡献，难度很大

国产新语言latte，基本上1234他都做了，2还在完善，所以我们尝试使用并没有那么大的困难，就可以试试了

编辑于 2017-12-26

▲ 4 ▼ 添加评论 分享 收藏 感谢 收起

NetReg
RD

2 人赞同了该回答

个人感觉一个主要原因是JVM有很多成熟的framework，opensource的东西，这些新语言就能利用这些...

发布于 2012-01-05

▲ 2 ▼ 添加评论 分享 收藏 感谢

贾明宇
软件架构师，自由主义者

1 人赞同了该回答

没必要重新造轮子，况且你短时间内顶多造个木轮子，人家这米其林现成的。

发布于 2016-01-26

▲ 1 ▼ 添加评论 分享 收藏 感谢

知乎用户
这个世界是一个智慧的创造

只有这样才能抢占Java的世界

发布于 2013-11-26

▲ 0 ▼ 添加评论 分享 收藏 感谢

张永飞
Java开发

1 人赞同了该回答

class文件可以实现语言无关性，在设计时就考虑了不只针对Java而且可以从其他语言编译成class文件。jvm解释或编译执行class文件可以实现平台无关性，可以跨平台

发布于 2015-08-19

▲ 1 ▼ 添加评论 分享 收藏 感谢

黑魔法师
Android程序员

8 人赞同了该回答

你造我们有个教授用大半辈子写的成果。结果呢，win7 断代升级 全都不能用了。这什么样的体验

发布于 2015-08-19

▲ 8 ▼ 9 条评论 分享 收藏 感谢

王科玮

1 人赞同了该回答

放一个传送门:看完就知道了
CSDN 视频不知道有没有和RednaxelaFX的ppt配套

编辑于 2015-08-19

▲ 1 ▼ 添加评论 分享 收藏 感谢

蜗牛
哈哈

语言和执行平台分开，这就是当初设计的初衷

发布于 2017-12-26

▲ 0 ▼ 添加评论 分享 收藏 感谢

令狐冲
纽扣学院创始人

因为实践证明JVM在工程领域的成功；而且JVM的开放性，也打造了完善的生态体系；所以，很多语言可以在语法层面重新定义语言，而不需要再自己造轮子了

编辑于 2017-07-16

▲ 0 ▼ 添加评论 分享 收藏 感谢

Alan Wang
IT民工

时至今日 很多犇人看不下去JVM被java糟蹋，纷纷在jvm上做起新语言或者移植一些popular的语言

发布于 2016-04-19

▲ 0 ▼ 添加评论 分享 收藏 感谢

任大大Gordon
过了CFA三级的运维

JVM把纷乱的后宫都统一了，我用C++可以写自己的语言而不用管理后端跨平台。

发布于 2016-04-19

▲ 0 ▼ 添加评论 分享 收藏 感谢

机智的老张
星星爱好者

第一位说的很全面，但是我想这个问题或许还有一种答案。
如果重新开发语言运行环境会花掉大量的时间，现在语言飞速发展投入精力去开发自己语言的运行

环境之后，可能会面临被同一时间使用成熟运行环境的语言所压制，也就是说，你的想法已经过时了。
确保时间节点上的优势，聪明人往往会选择依赖现有成熟案例，例如CLR，JVM等。
在这写选择里，有两点，开放和免费尤其重要，毕竟先投入风险巨大。所以大家都选择了JVM。

发布于 2013-02-27

▲ 0 ▼ 添加评论 分享 收藏 感谢

知乎用户

这种问题很难回答,选择哪个vm要根据自己的需求.笼统的说除了跨平台，jvm的优点还有:
1.成熟稳定,经过这么多年,相关的功能和文档都非常的完善。特别是后来引入了对脚本语言的支持，很多新语言选择了jvm.
2.丰富的java代码资源可以调用。
3.性能还是比较好的.

编辑于 2012-01-03

▲ 0 ▼ 2 条评论 分享 收藏 感谢