CommunityOverCode

THE ASF CONFERENCE

GraalVM static compilation in web container application

Zihao RAO(饶子昊)
Alibaba Cloud R & D engineer

THE APACHE® SOFTWARE FOUNDATION

CONTENTS

THE APACHE® SOFTWARE FOUNDATION

CommunityOverCode

THE ASF CONFERENCE

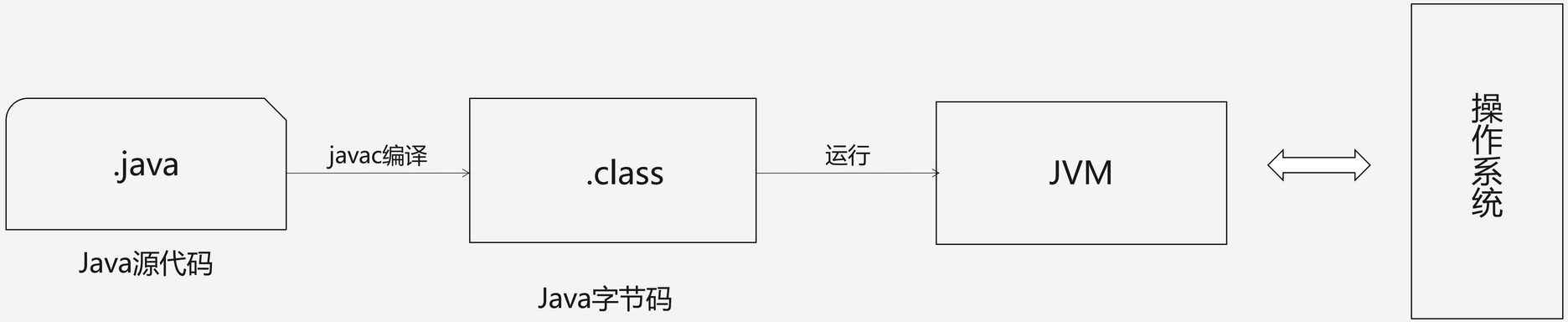
Part 01

Java Web容器应用运行回顾

THE APACHE® SOFTWARE FOUNDATION

Java Web 应用启动运行过程

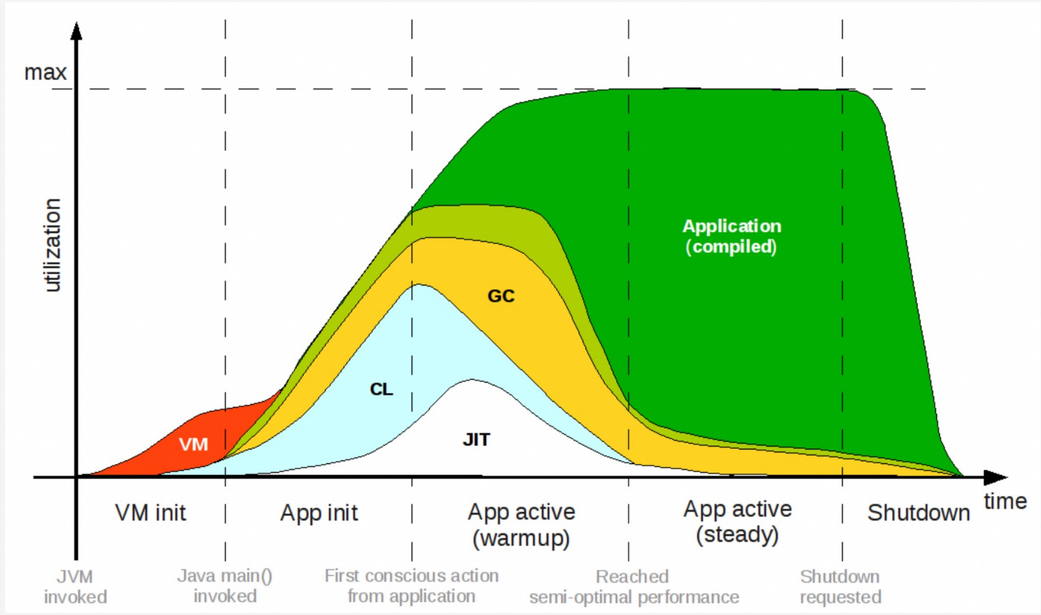
在正式静态编译技术之前，首先简单回顾一下一个 Java Web 应用程序从开发、打包、编译执行的过程：



Java Web应用编译执行过程

Java Web 应用的局限性

一个 Java Web 程序的执行生命周期如下图所示，分为JVM初始化、应用程序初始化、应用预热、应用稳定执行和关闭5个部分：



Java应用生命周期

图片来自：https://shipilev.net/talks/j1-Oct2011-21682-benchmarking.pdf

CommunityOverCode

THE ASF CONFERENCE

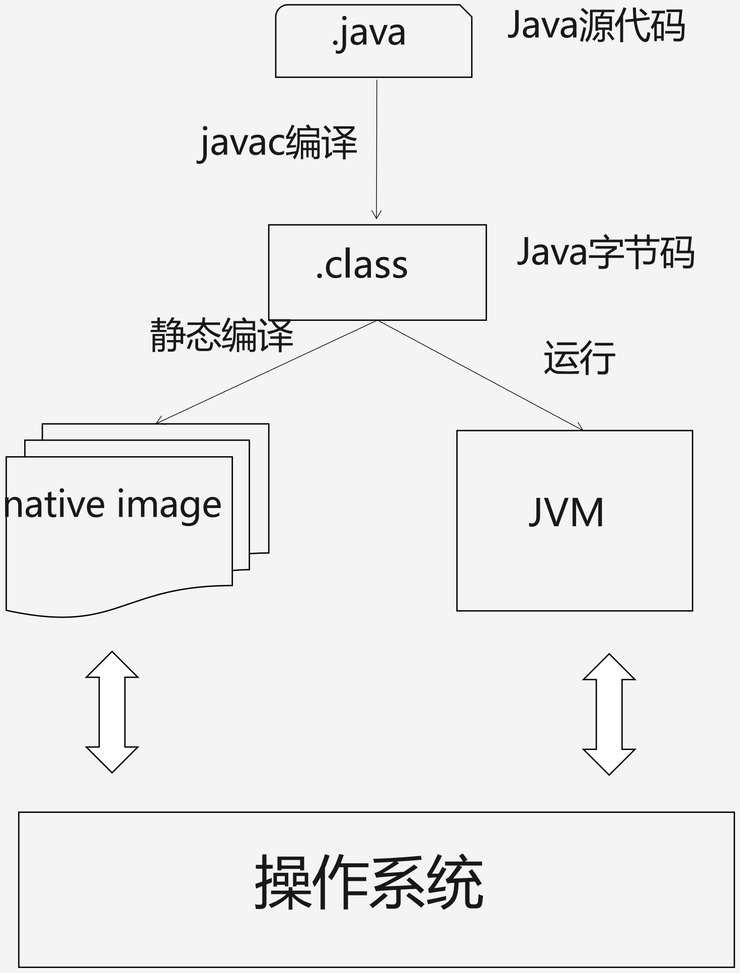
Part 02

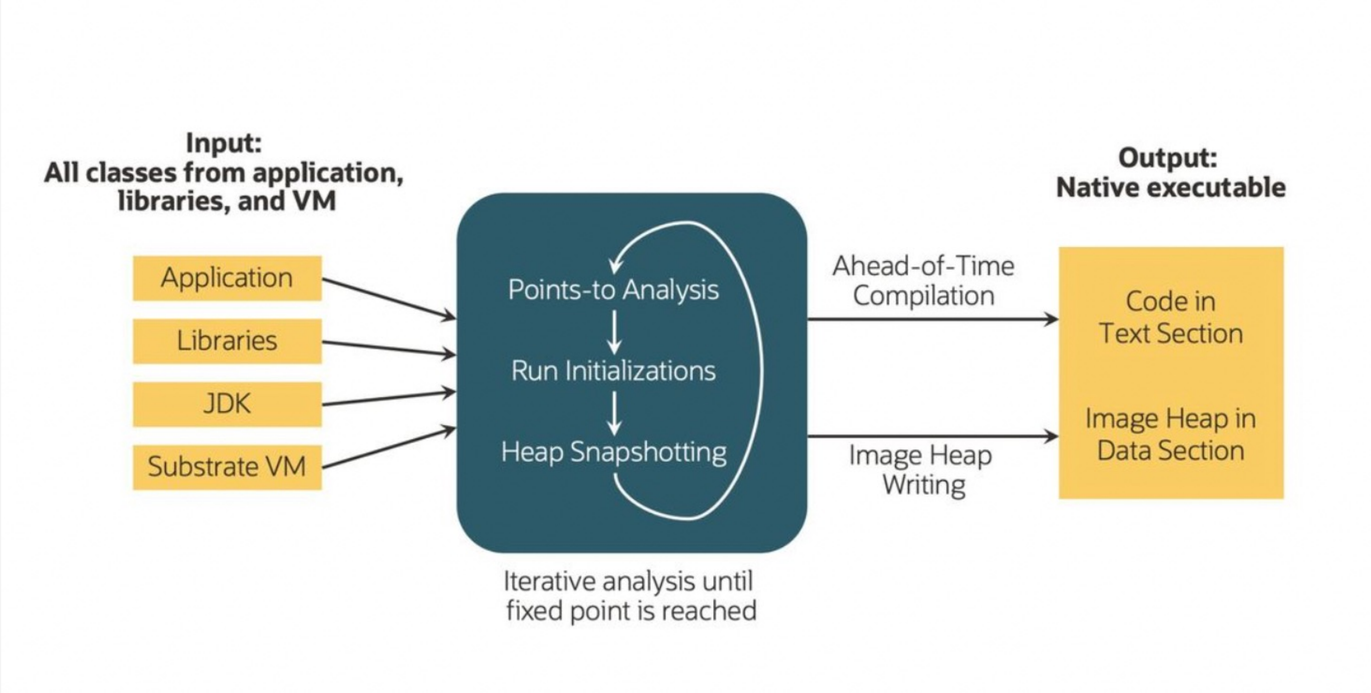
静态编译技术

THE APACHE® SOFTWARE FOUNDATION

静态编译

静态编译是相对于Java的动态编译而言的，其是指将Java程序的字节码在单独的离线阶段编译为汇编语言。静态编译过程相比于动态编译过程如下图所示：



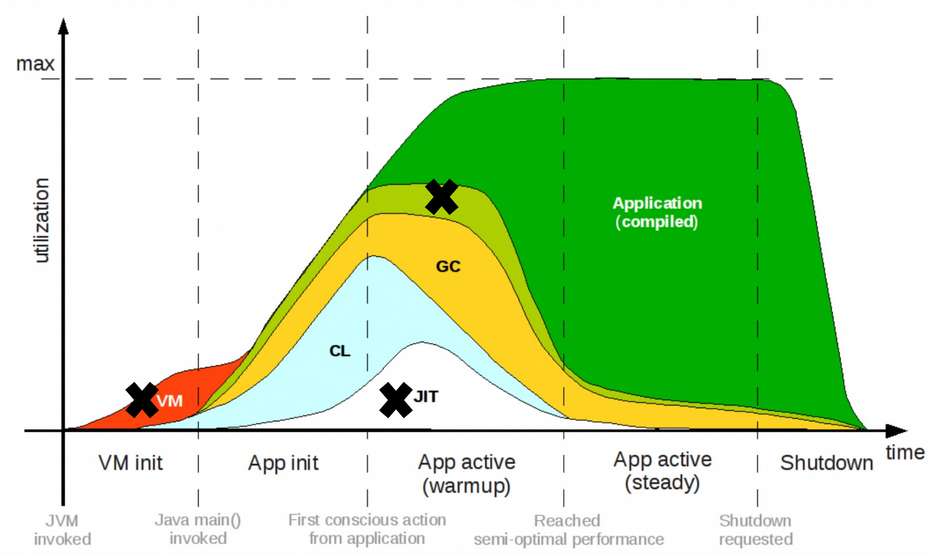


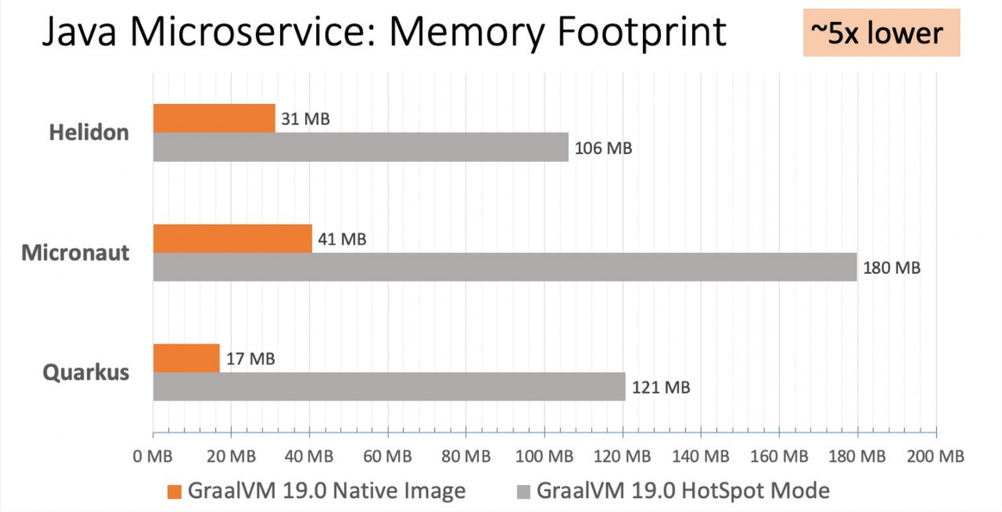
图片来自：GraalVM静态编译原理与应用、https://www.infoq.cn/article/bd4xlkdi8emx4ysigt9t

静态编译优势

与传统基于JVM的Java应用程序运行方式，采用静态编译技术能给Java应用程序带来如下优势：

- 消除冷启动：没有了程序解释执行、虚拟机初始化等步骤，冷启动问题得到大幅度改善。
- 内存占用率大幅降低：静态编译做了大量优化，并且没有了虚拟机运行时内存占用，内存占用得到大幅降低。



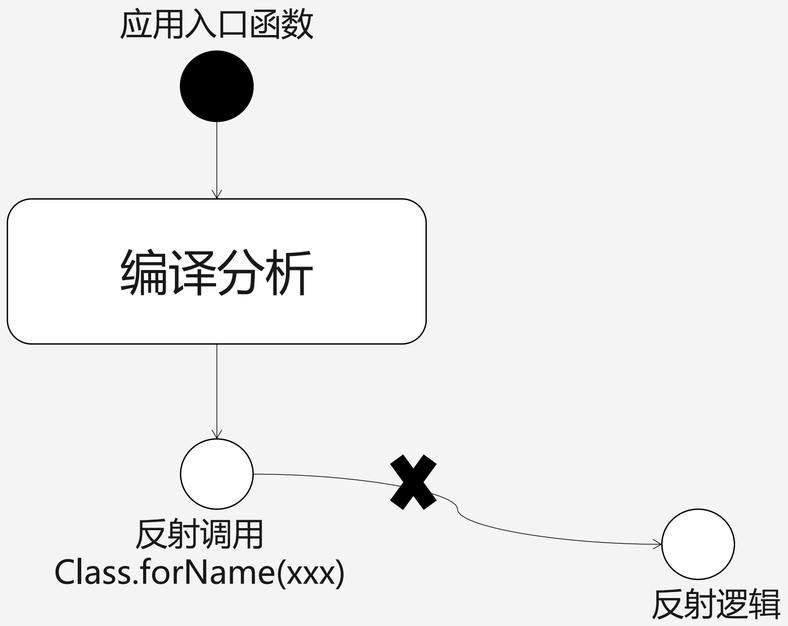


图片来自：https://learn.lianglianglee.com/

静态编译影响

采用静态编译，程序运行阶段直接使用的是提前编译好的二进制可执行文件，会对Java生态的以下方面产生影响：

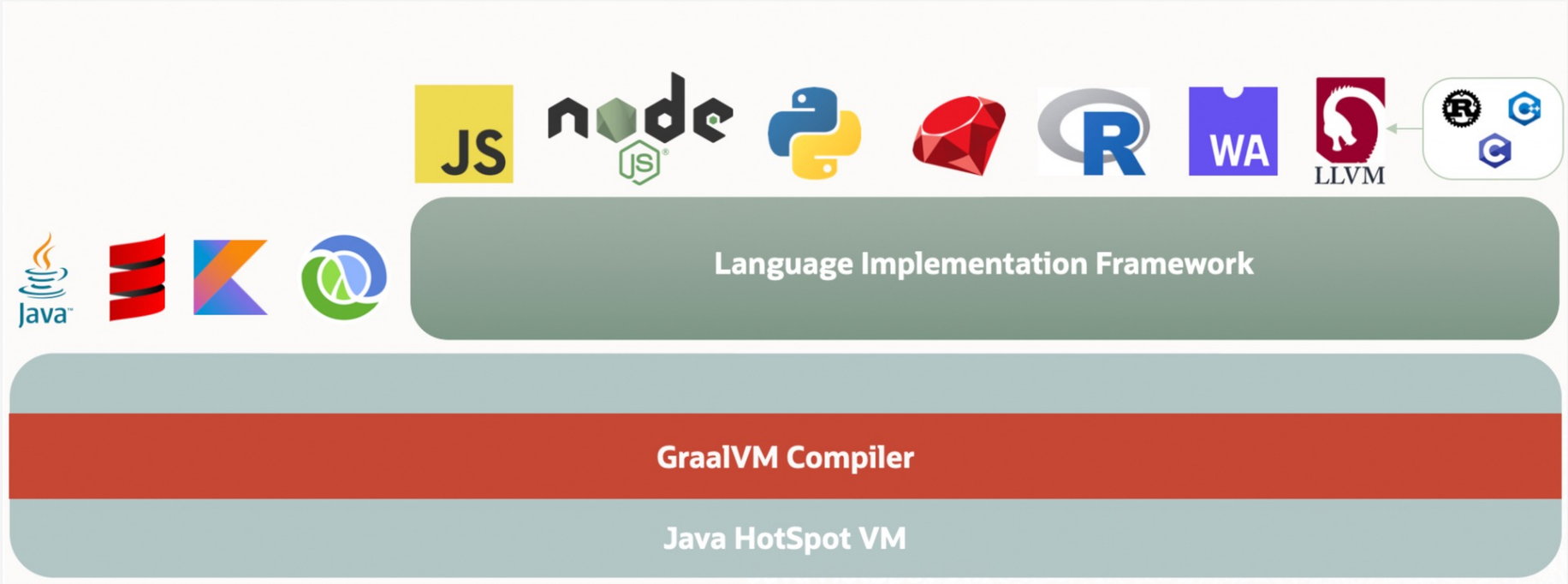
- 动态特性：动态类加载、反射、动态代理、JNI和序列化等动态特性不再完全支持。
- 平台无关性：由于没有了JVM和平台无关的字节码文件，Java平台引以为傲的平台无关性不再具备。
- 生态工具：由于没有了JVM和平台无关的字节码文件，原来Java生态的监控、调试和Agent扩展等功能都不再生效。




```
String className = "com.example.springboot3.Engineer";
Class<?> clazz = Class.forName(className);
Method method = clazz.getDeclaredMethod( name: "sayHello");
method.setAccessible(true);
return (String) method.invoke(clazz.newInstance());
```

静态编译之GraalVM

GraalVM是Oracle推出的基于Java开发的开源高性能多语言运行时平台。其目标是消除不同语言之间的壁垒，在统一的运行时平台上实现跨语言的程序交互。

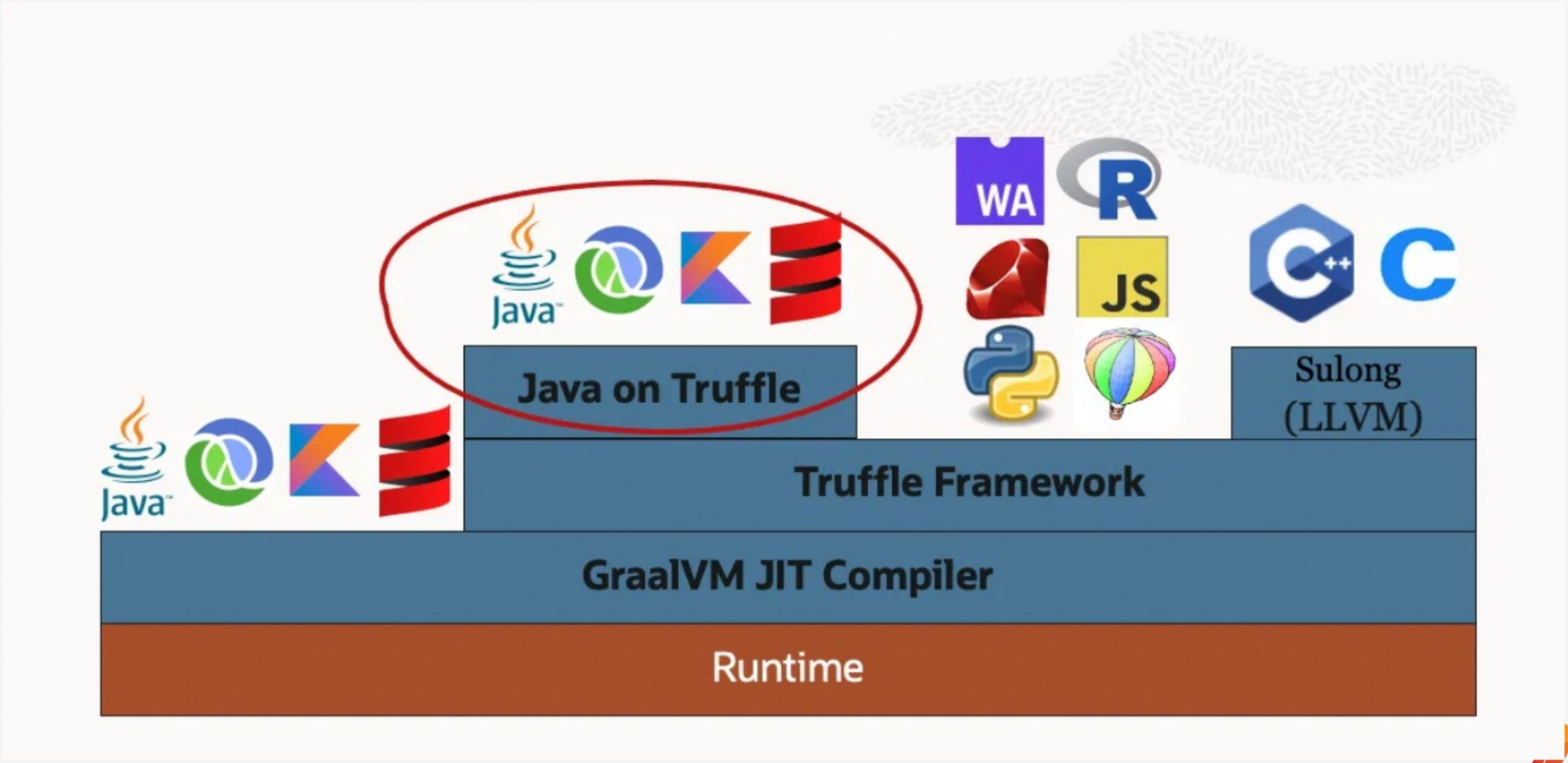


图片来自：https://www.graalvm.org/latest/docs/introduction/

THE APACHE SOFTWARE FOUNDATION

静态编译之GraalVM

Truffle是GraalVM中的解释器实现框架，开发人员可以使用Truffle提供的API快速用Java实现一种语言的解释器。GraalVM实现静态编译能力的编译器就是GraalVM JIT Compiler。静态编译框架和运行时由Substrate VM子项目实现，兼容OpenJDK运行时实现，提供了原生镜像程序运行时的异常处理、同步调度、线程管理、内存管理等功能。



图片来自：https://medium.com/graalvm/java-on-truffle-going-fully-metacircular-215531e3f840

CommunityOverCode

THE ASF CONFERENCE

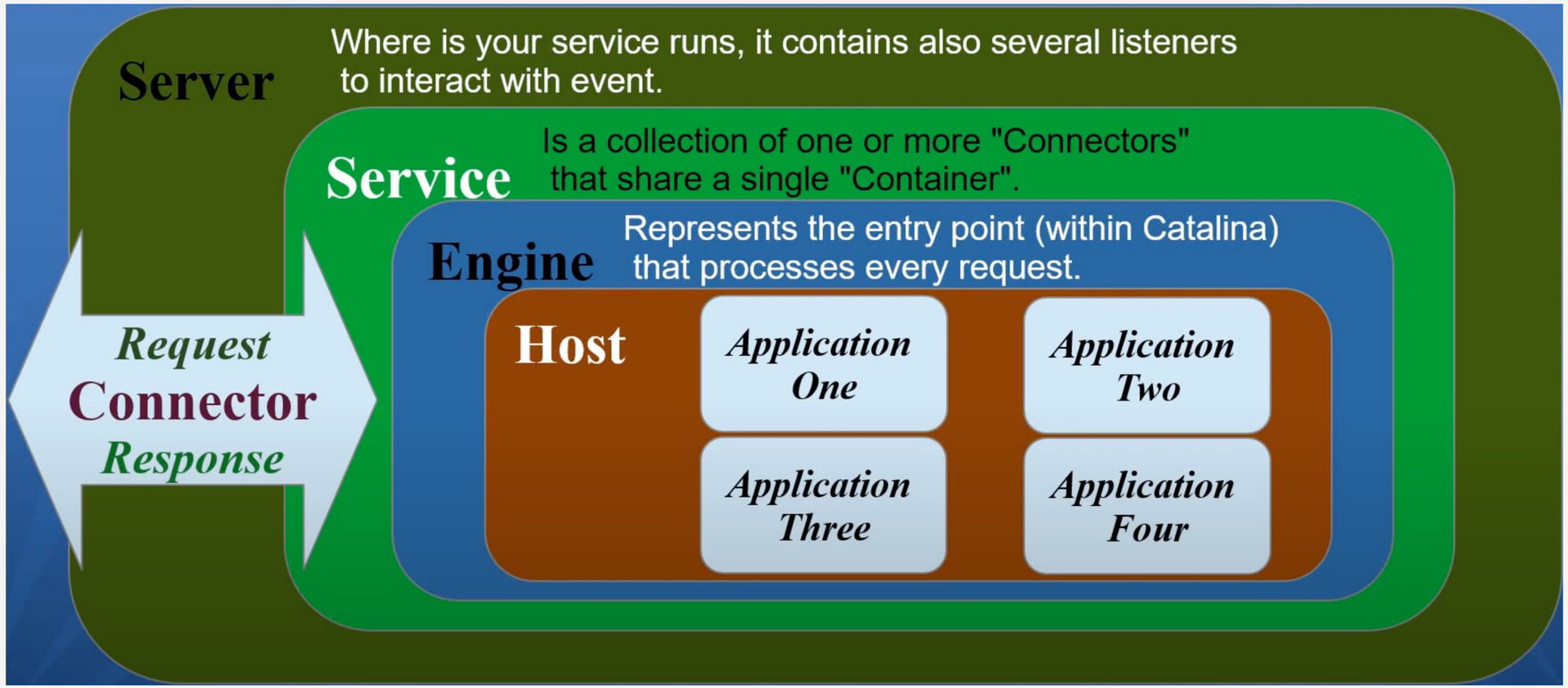
Part 03

静态编译助力Java Web容器云原生化

THE APACHE® SOFTWARE FOUNDATION

Apache Tomcat介绍

Apache Tomcat 作为应用广泛的 Web 容器运行中间件，其架构图如下图所示：



Tomcat架构图

Picture by: Ahmed Samy- software engineer

THE APACHE SOFTWARE FOUNDATION

部分Tomcat容器静态编译效果

采用 GraalVM 静态编译技术基于 Tomcat 容器运行时的 Spring Cloud Alibaba 微服务 Web 应用，所有核心能力在启动速度和内存占用率方便都有显著改善。例如，Nacos服务启动速度提升了近10倍、内存占用率降低为原来1/3。

	Nacos			RocketMQ		Sentinel		Seata
应用类型	服务注册	服务消费	监听配置	消息发送	消息消费	客户端限流	客户端降级	Seata 客户端
启动速度（GraalVM）	0.339s	0.479s	0.182s	0.236s	0.348s	0.534s	0.521s	2.999s
启动速度（JVM）	4.161s	4.763s	2.529s	3.138s	4.392	3.065s	3.485s	8.611s
内存占用（GraalVM）	57.9M	59.0M	50.1M	91.0M	68.2M	61.0M	60.7M	112.4M
内存占用（JVM）	141.4M	129.8M	122.7M	217.5M	181.8M	152.1M	151.4M	263.6M

上述测试代码样例来自Spring Cloud Alibaba项目中的examples模块，4c16g Mac环境，每组数据测试3次取平均，具体数据因机器不同可能会有差异。

CommunityOverCode

THE ASF CONFERENCE

Thanks

WWW.COMMUNITYOVERCODE.ORG

THE APACHE® SOFTWARE FOUNDATION