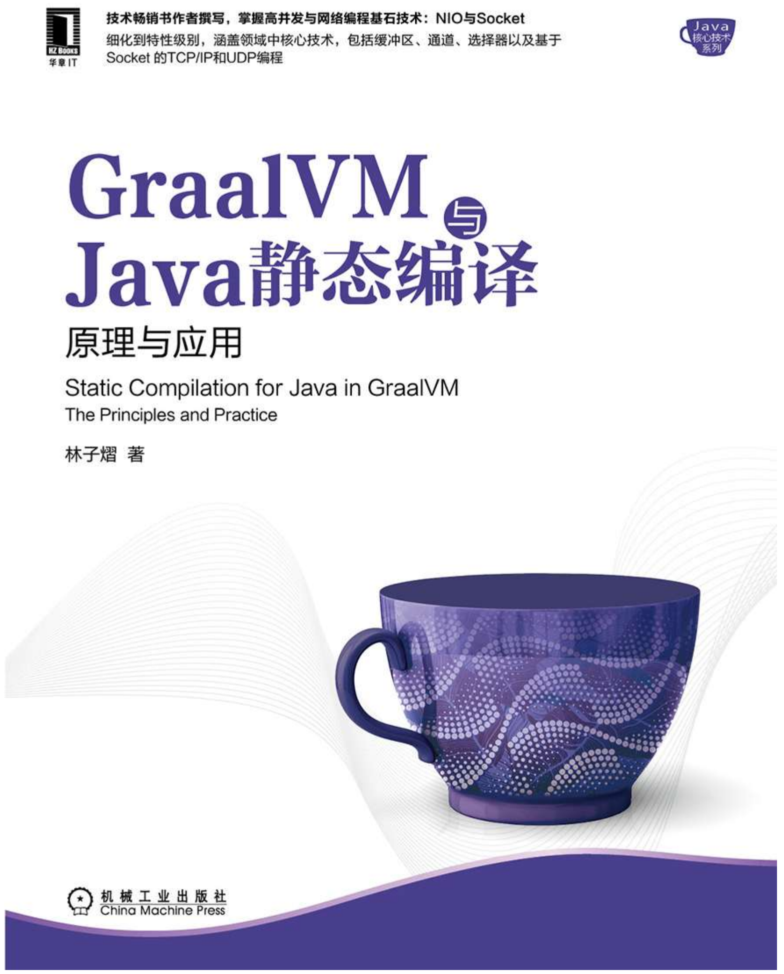
Implement auto instrumentation under GraalVM static compilation on OTel Java Agent

Huxing ZHANG & Zihao RAO & **Ziyi LIN**, Alibaba Cloud

2024 Java生态发展论坛

关于我 林子熠

- 阿里云基础产品事业部编译器JVM团队
- 上海交通大学博士、CCF系统专委会执行委员
- ACM SIGSOFT杰出论文奖获得者
- Apache社区committer
- GraalVM社区的活跃提交者
- 《GraalVM与Java静态编译：原理与应用》作者



华章IT

技术畅销书作者撰写，掌握高并发与网络编程基石技术：NIO与Socket
细化到特性级别，涵盖领域中核心技术，包括缓冲区、通道、选择器以及基于Socket 的TCP/IP和UDP编程

Java 核心技术系列

GraalVM与
Java静态编译
原理与应用

Static Compilation for Java in GraalVM
The Principles and Practice

林子熠 著

机械工业出版社
China Machine Press

01

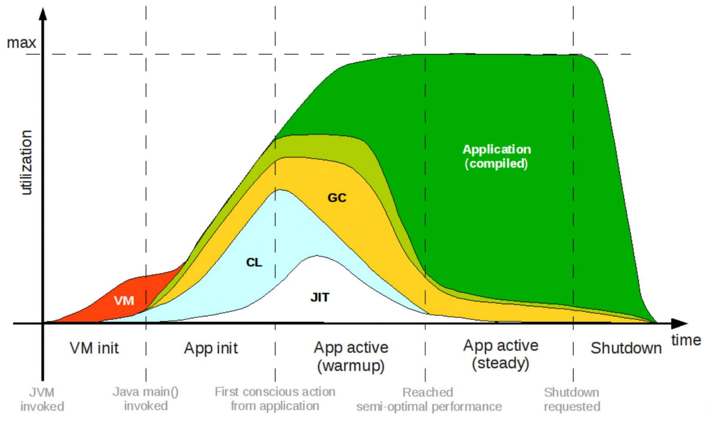
Background

Challenges for modern Java applications

Slow startup

High memory overhead

Lifecycle of Java applications: VM init, App init, warmup, App active and shutdown:



Lifecycle of Java apps

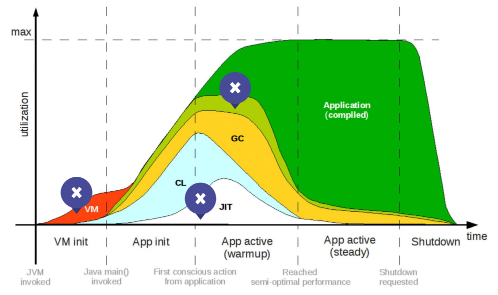
Picture by: https://shipilev.net/talks/j1-Oct2011-21682-benchmarking.pdf

Introduction of GraalVM native image

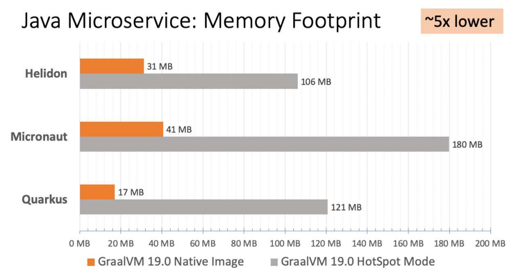
Compared to JVM-based environments, GraalVM offers the following advantages

Enhanced startup speed: By eliminating VM init, JIT, and interpretation overhead, the startup time is significantly reduced

Reduced memory overhead: By removing the memory footprint associated with the VM and applying numerous optimizations, memory usage is significantly reduced



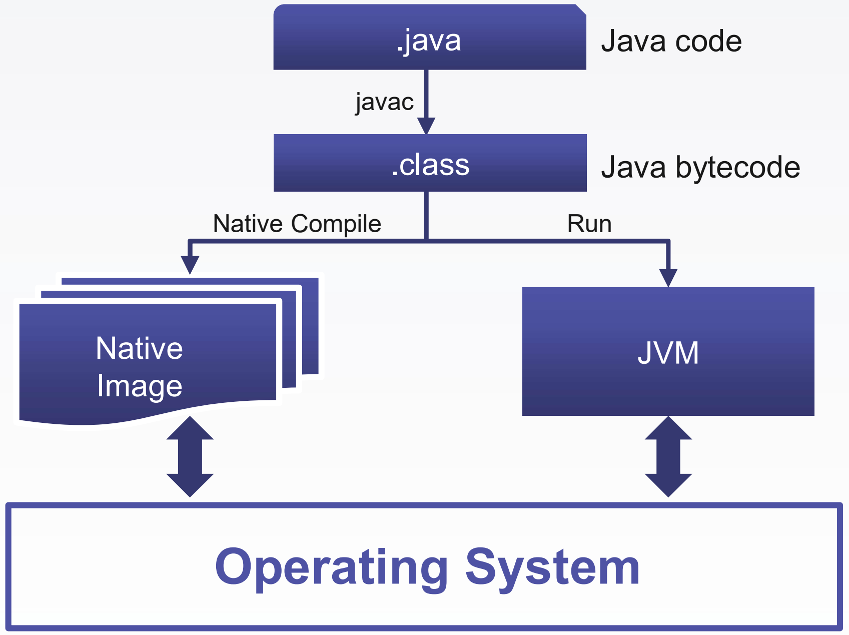
Lifecycle of Java apps under GraalVM



Improvements of different frameworks

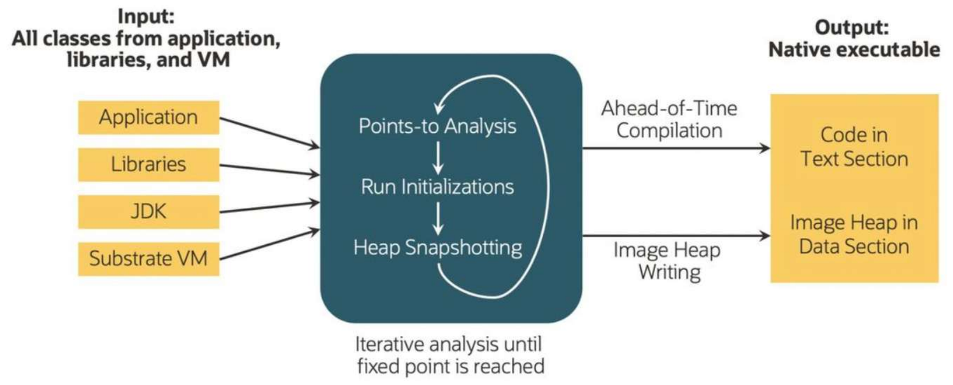
Picture by: https://shipilev.net/talks/j1-Oct2011-21682-benchmarking.pdf

GraalVM native image compilation process



Comparation of JVM and native compilation

The process of native compile:



Process of native compilation

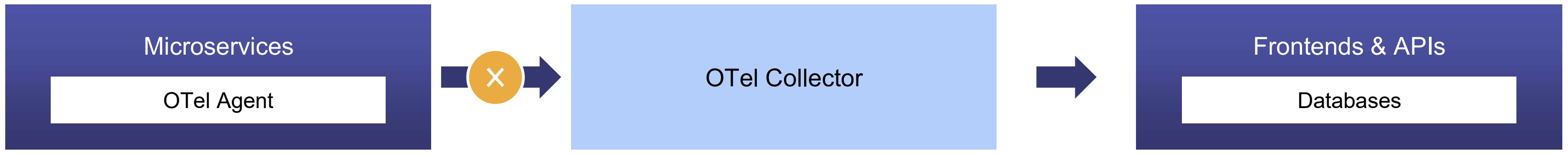
Picture by: https://www.infoq.com/articles/native-java-graalvm/

Impacts of GraalVM on the Java Ecosystem

Dynamic Features: Dynamic class loading, reflection, dynamic proxies, JNI, and serialization are no longer fully supported

Platform Independence: Without the JVM and bytecode, the platform independence that is a hallmark of the Java platform is no longer available

Ecosystem Tools: The original Java ecosystem tools for monitoring, debugging, and Java Agents are ineffective without the JVM and bytecode



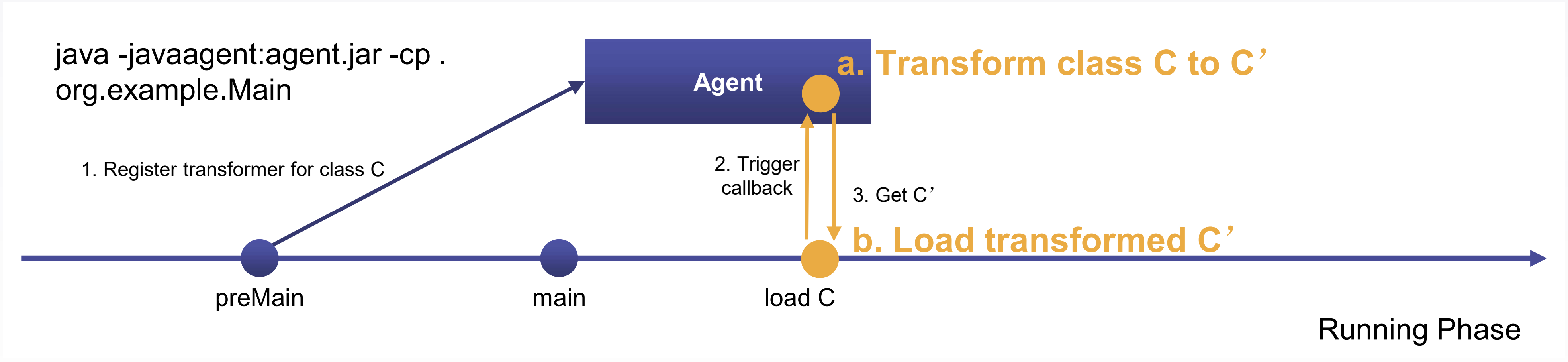
Impact of GraalVM in observability

02

Solution

Idea to instrument under GraalVM

Java Agent work process:


java -javaagent:agent.jar -cp . org.example.Main
Agent
a. Transform class C to C'
1. Register transformer for class C
2. Trigger callback
3. Get C'
b. Load transformed C'
preMain
main
load C
Running Phase

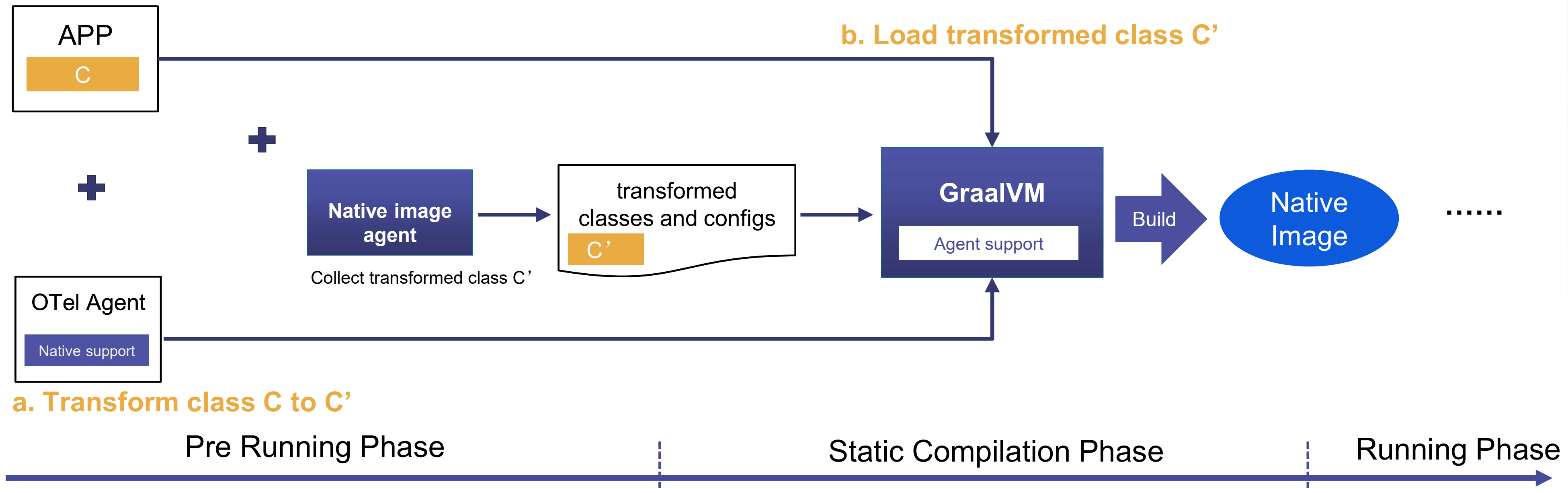
Runtime problems turn to build problems:

a. How to transform target classes before building?

b. How to let transformed classes take effect at build time?

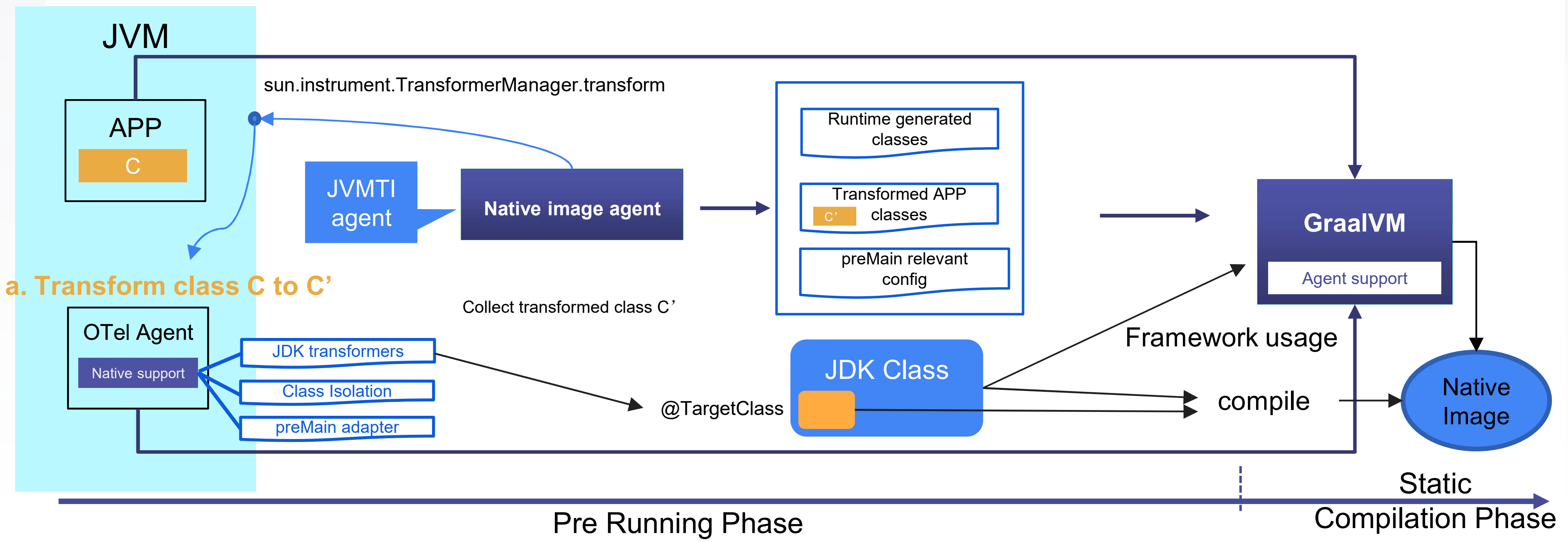
Overall design

Static Instrumentation



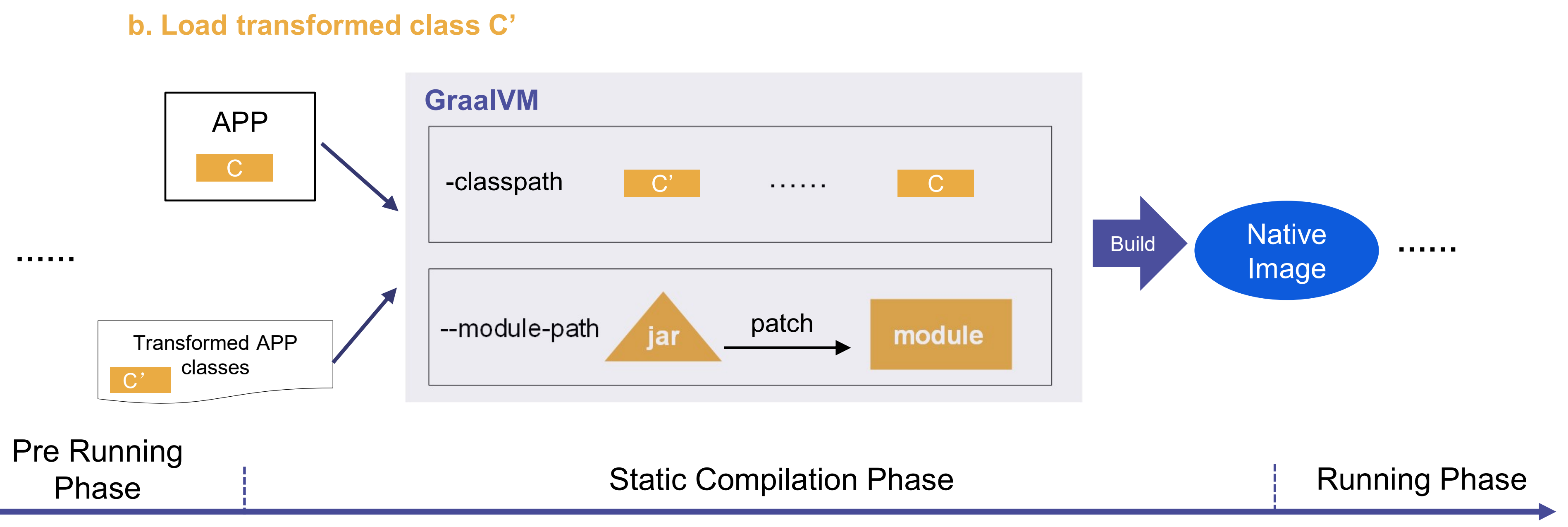
Transform and record classes

Implemented an interceptor in native image agent to collect transformed classes:



How to Apply Transformed Classes

Load transformed classes by -classpath and --module-path:



03

Demonstration

Demonstration

Experimental Result

Comparison of startup speed and memory overhead: JVM vs. GraalVM native image with Java Agent

	Spring Boot	Kafka	Redis	MySQL
Startup Speed (JVM)	7.541s	11.323s	10.717s	8.116s
Memory Overhead (JVM)	402MB	408MB	420MB	394MB
Startup Speed (GraalVM)	0.117s （-98%）	0.168s （-98%）	0.152s （-98%）	0.119s （-98%）
Memory Overhead (GraalVM)	96MB （-75%）	141MB （-65%）	128MB （-69%）	107MB （-73%）

32 vCPU/64 GiB/5 Mbps

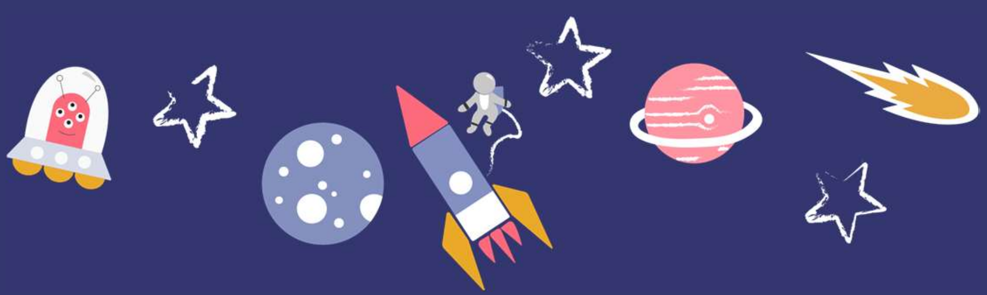
04

Future works

Future works

In the future, we plan to focus on the following aspects:

1. Conduct comprehensive test cases over multiple signals(metrics, trace, logs, and etc).
2. Consolidate the pre-running phase and the native compilation phase into a unified phase to ensure transformed classes are universally collected.



Thanks
Q&A

Motivation:
Support Agent Instrumentation in GraalVM native image

Insight:
Turn a runtime problem to a compilation problem

Relevant Pull Requests:
(1) Native Support in OTel Java Agent:
https://github.com/open-telemetry/opentelemetry-java-instrumentation/pull/11068

(2) Agent Support in GraalVM:
https://github.com/oracle/graal/pull/8077