Java **Module** & **A**head **O**f **T**ime Compilation

Battle of Efficiency

Luram Archanjo

Who am I?

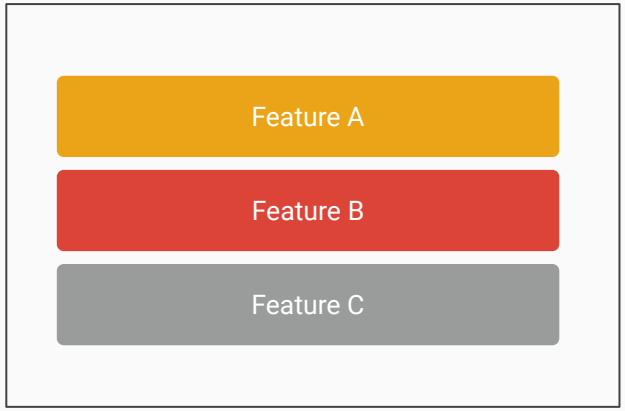
- Software Engineer at Sensedia
- MBA in Java projects
- Java and Microservice enthusiastic

Agenda

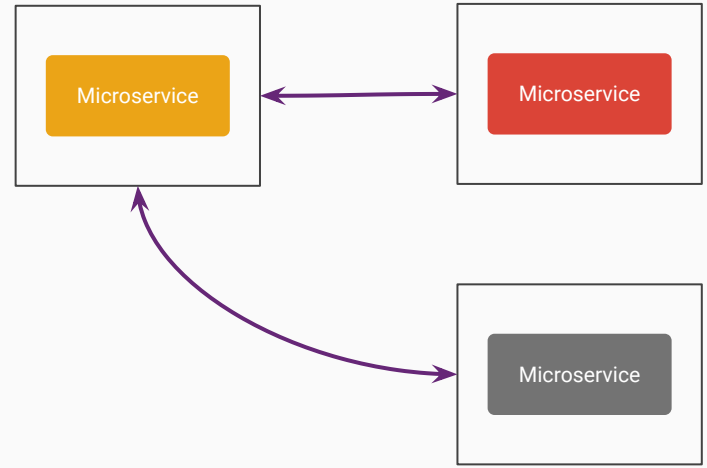
- Microservices
- Java **Module**
- **A**head **O**f **T**ime Compilation (AOT)
- **J**ust **I**n **T**ime Compilation (JIT)
- Native Image
- Questions

Moving to Microservices

Monolith

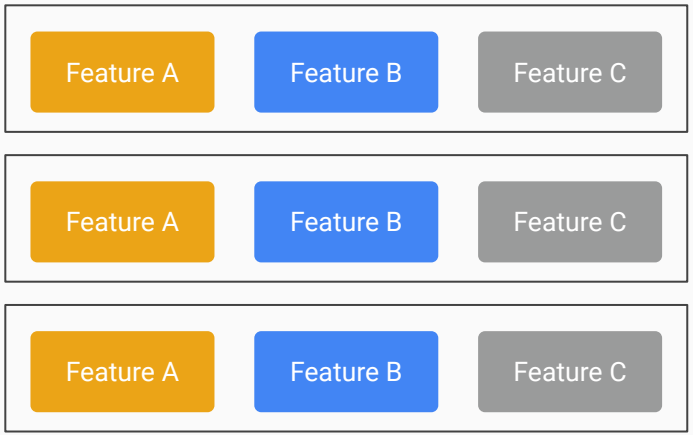


Microservices



Scalability

Monolith Scalability



Microservices Scalability

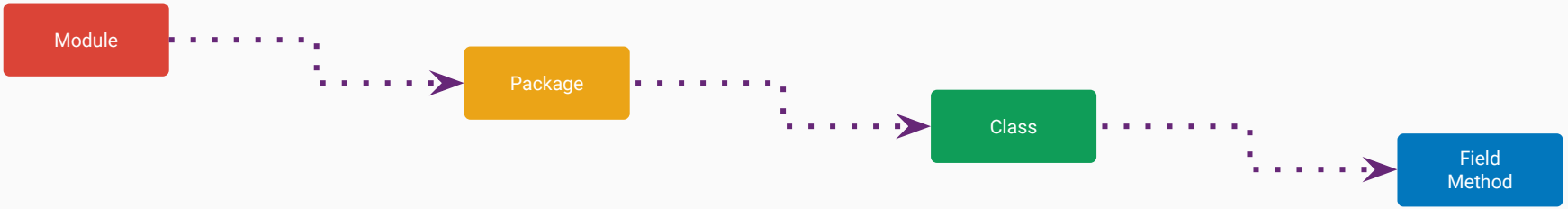
Our resources are **finite**!

How to use **less resources** using Java Language?

Java **Module**

Java **Module**

Modularity adds a **higher level of aggregation** above packages. The key new language element is the module - a uniquely named, reusable group of related packages, as well as resources and a module descriptor specifying.

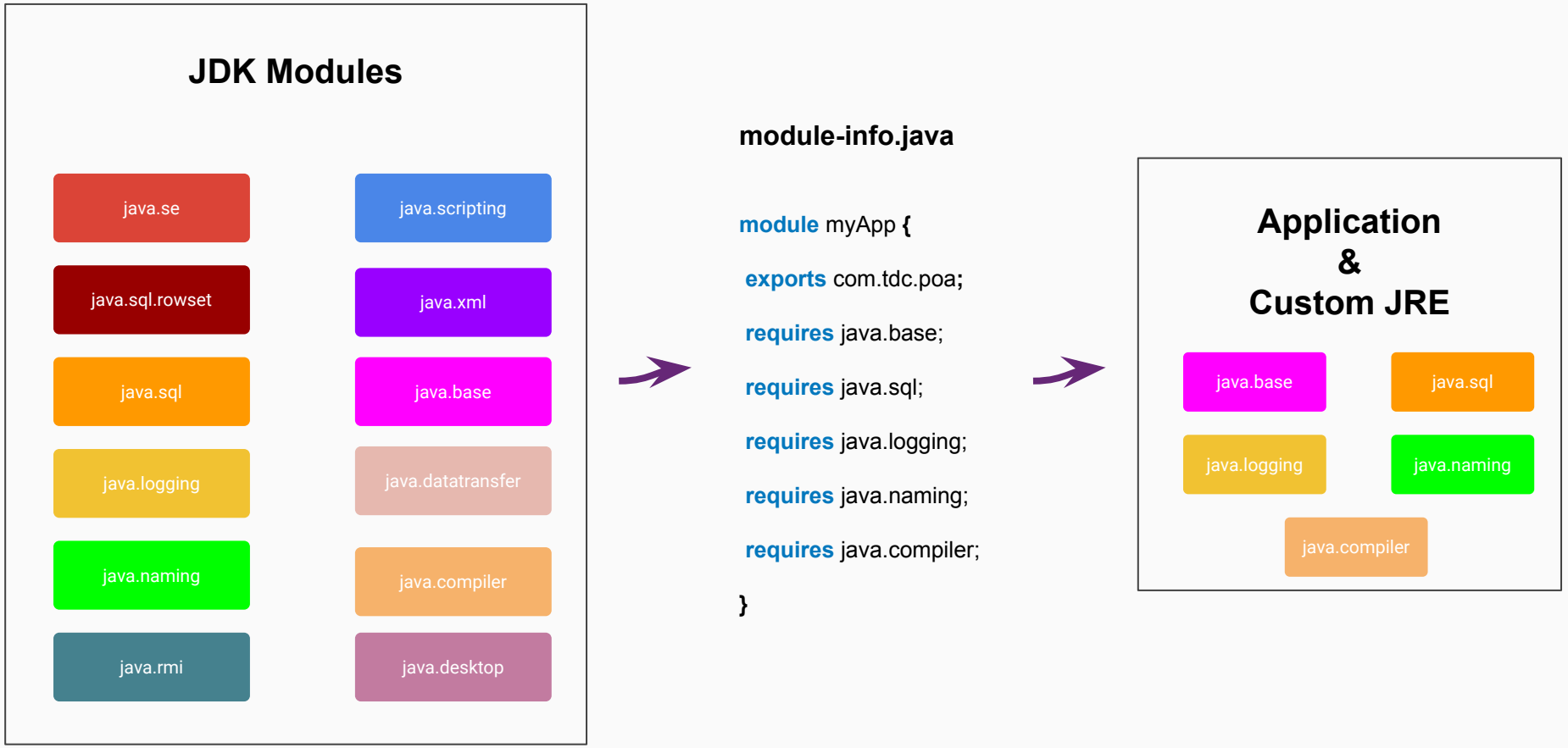


According to **JSR 376**, the key goals of modularizing the Java SE platform are:

- Reliable Configuration
- Strong Encapsulation
- Greater Platform Integrity
- **Scalable Java Platform**
- **Improved Performance**

Source: https://jcp.org/en/jsr/detail?id=376 & https://www.oracle.com/corporate/features/understanding-java-9-modules.html

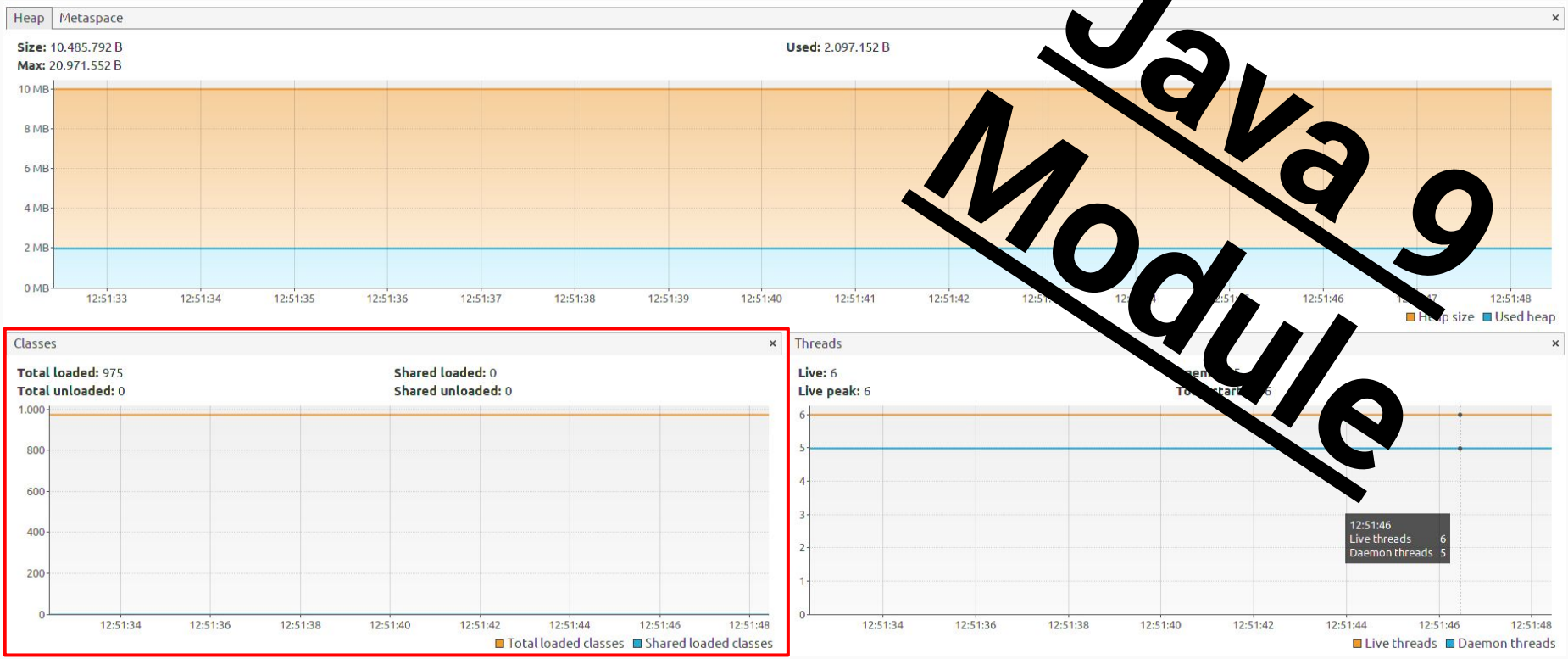
Java Module


JDK Modules
java.se
java.scripting
java.sql.rowset
java.xml
java.sql
java.base
java.logging
java.datatransfer
java.naming
java.compiler
java.rmi
java.desktop
module-info.java
module myApp {
exports com.tdc.poa;
requires java.base;
requires java.sql;
requires java.logging;
requires java.naming;
requires java.compiler;
}
Application
&
Custom JRE
java.base
java.sql
java.logging
java.naming
java.compiler

What are the **results** of using Java **Modules**?

What are the **results** of using Java **Modules**?

What are the **results** of using Java **Modules**?



Java 9 Module

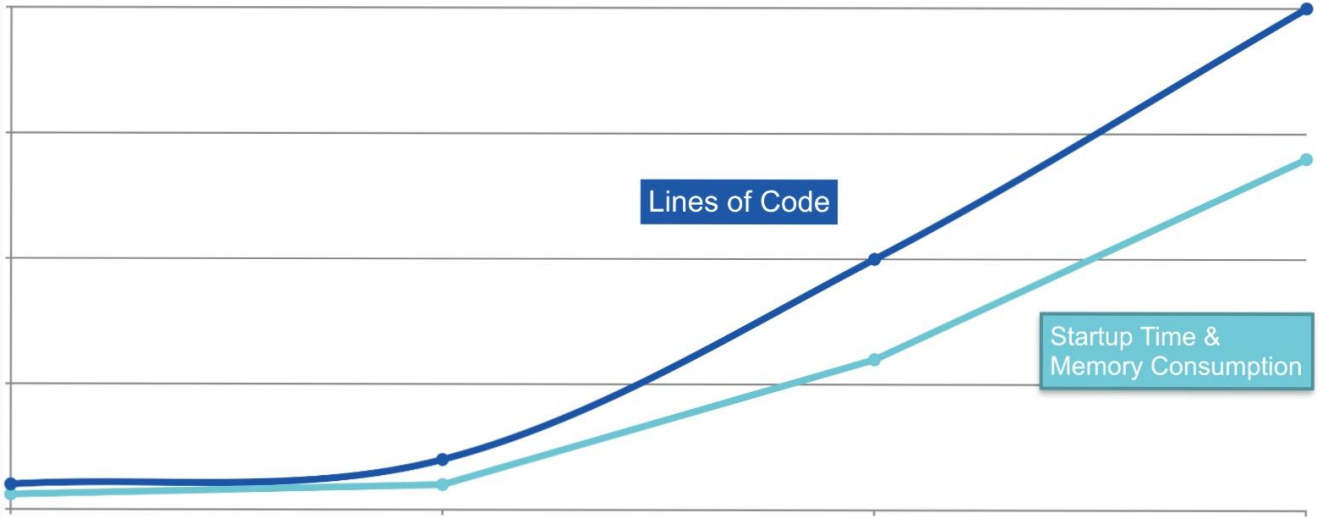
Less classes, functions and dependencies **are not enough**!

The **villain** of Java's resources is the **Reflection**

What are the results of using **Reflection**?

Spring is an amazing technical achievement and does so many things, but does them at **Runtime**.

- Reads the byte code of every bean it finds.
- Synthesizes new annotations for each annotation on each bean method, constructor, field etc. to support Annotation metadata.
- Builds Reflective Metadata for each bean for every method, constructor, field etc.



Is it possible to have the same **productivity** but without **Reflection**?

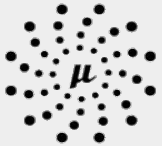
Yes, with **A**head **O**f **T**ime (AOT) Compilation

Ahead Of Time (AOT) Compilation

Ahead-of-time compilation (AOT compilation) is the **act of compiling a higher-level programming** language, or an intermediate representation such as Java bytecode, into a **native machine code** so that the resulting binary file can **execute natively**.

Web

Java



MICRONAUT

QUARKUS

Android

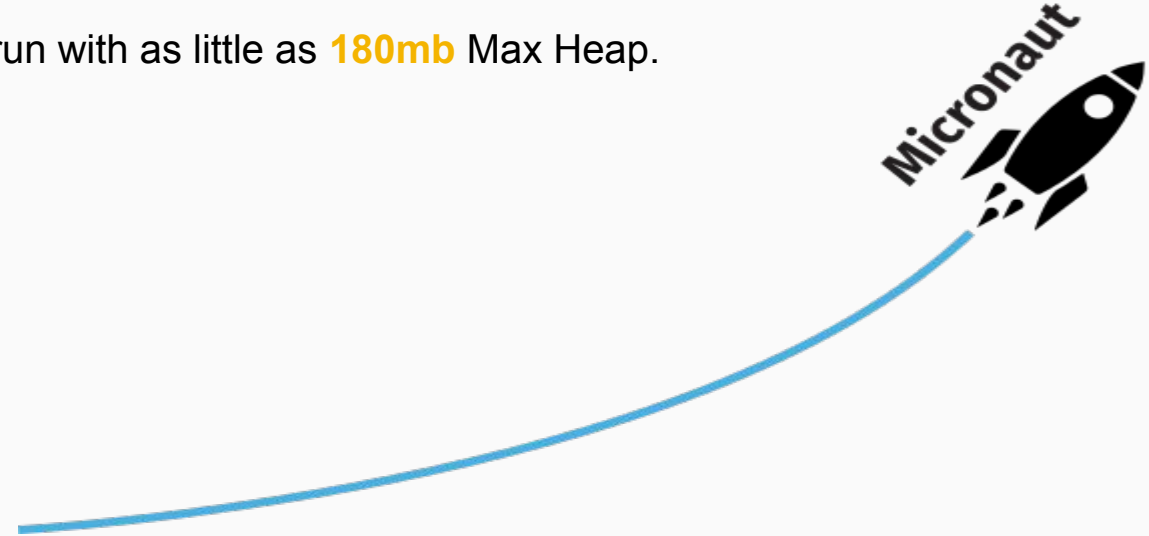
Google Dagger 2

What are the **results** of using **A**head **O**f **T**ime (AOT) Compilation?

What are the **results** of using **A**head **O**f **T**ime (AOT) Compilation?

Data from **Micronaut** website:

- Startup time around **a second**.
- All Dependency Injection, AOP and Proxy generation happens at **compile time**.
- Can be run with as little as **180mb** Max Heap.



What are the **results** of using **A**head **O**f **T**ime (AOT) Compilation?

Data from **Quarkus** website:

- Startup time around **two seconds**.
- All Dependency Injection, AOP and Proxy generation happens at **compile time**.
- Can be run with as little as **145mb** Max Heap.

QUARKUS

Is it possible **to improve** more?

Yes, with Just In Time (JIT) Compilation and

GraalVM™

GraalVM what is?

GraalVM is a universal virtual machine for running applications written in JavaScript, Python, Ruby, R, JVM-based languages like Java, Scala, Groovy, Kotlin, Clojure, and LLVM-based languages such as C and C++

- Native Image
- Embeddable

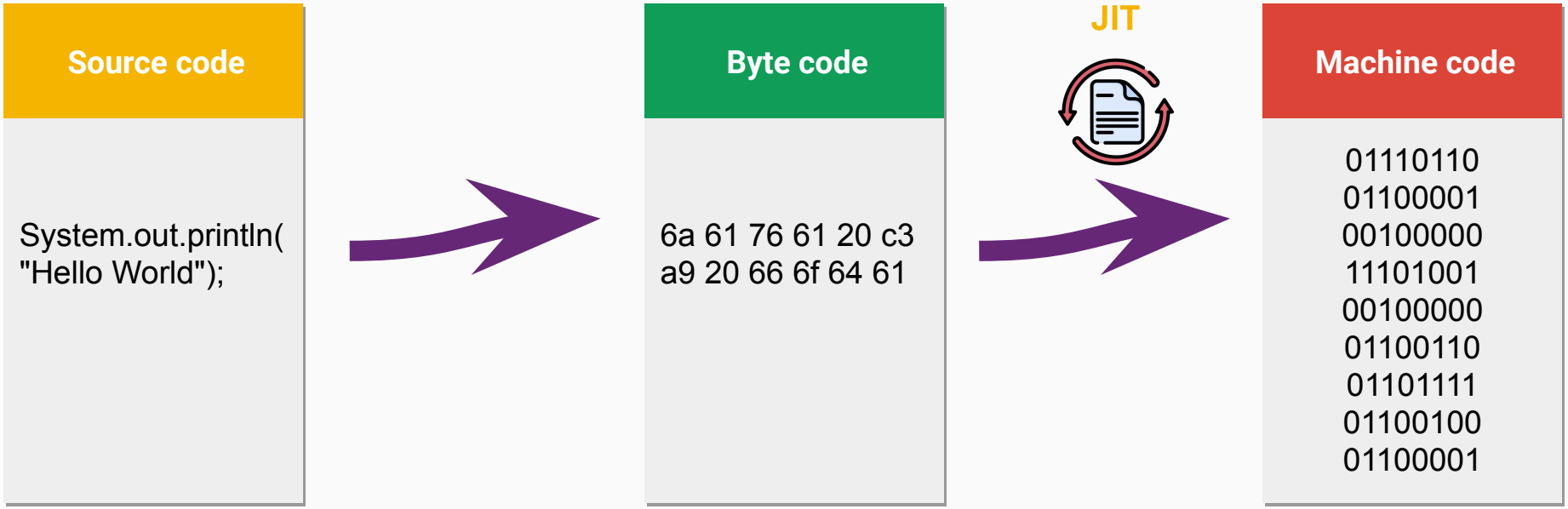
For Java Programs

For existing Java applications, GraalVM can provide benefits by running them faster, providing a faster **J**ust **I**n **T**ime (JIT) Compilation

Source: https://www.graalvm.org/docs/why-graal/

GraalVM Just In Time (JIT) Compilation

Just In Time (JIT) compilation is a way of executing computer code that involves **compilation during execution of a program**. It runs complex optimizations to generate **high-quality machine code**



What are the **results** of using **J**ust **I**n **T**ime (JIT) Compilation?

What are the **results** of using **J**ust **I**n **T**ime (JIT) Compilation?

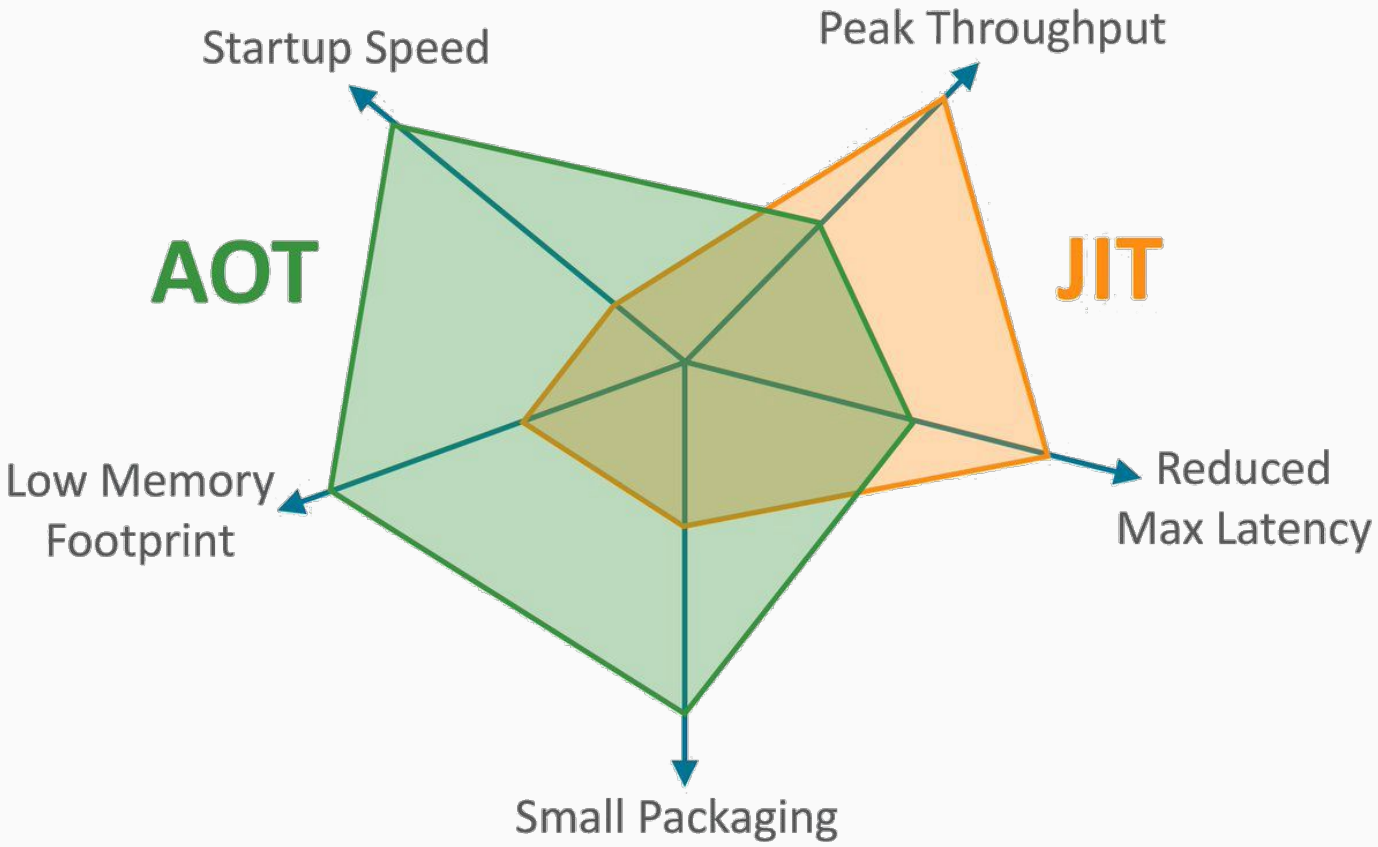
OpenJDK

Count	4589
Total	60.00 s
Slowest	3.79 s
Fastest	5.85 ms
Average	130.43 ms
Requests / sec	76.48

GraalVM™

Count	5815
Total	60.00 s
Slowest	2.36 s
Fastest	2.15 ms
Average	102.87 ms
Requests / sec	96.91

What are the **results** of using **J**ust **I**n **T**ime (JIT) Compilation?



Is it possible **to improve** more?

Yes, with **Native Image** and

GraalVM™

GraalVM Native Image, currently available as an **Early Adopter Technology**

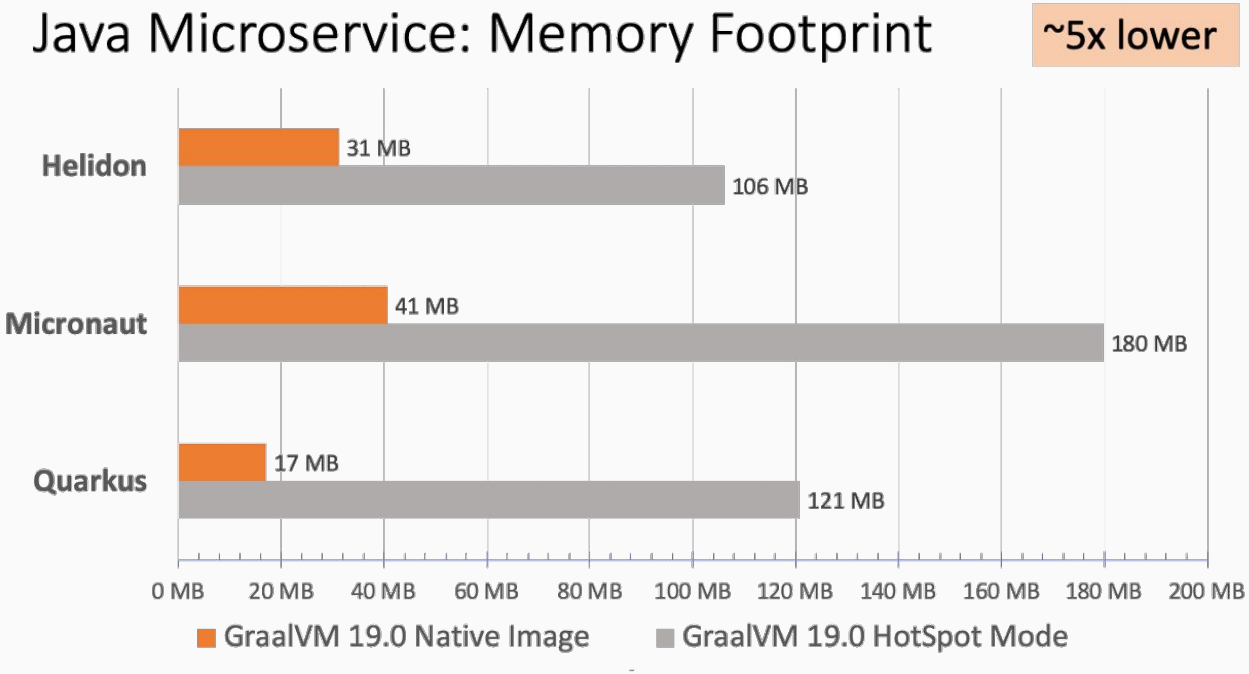
Native image works well when:

- Little or no runtime reflection is used.
- Limited or no dynamic classloading.

Source: https://www.graalvm.org/docs/why-graal/

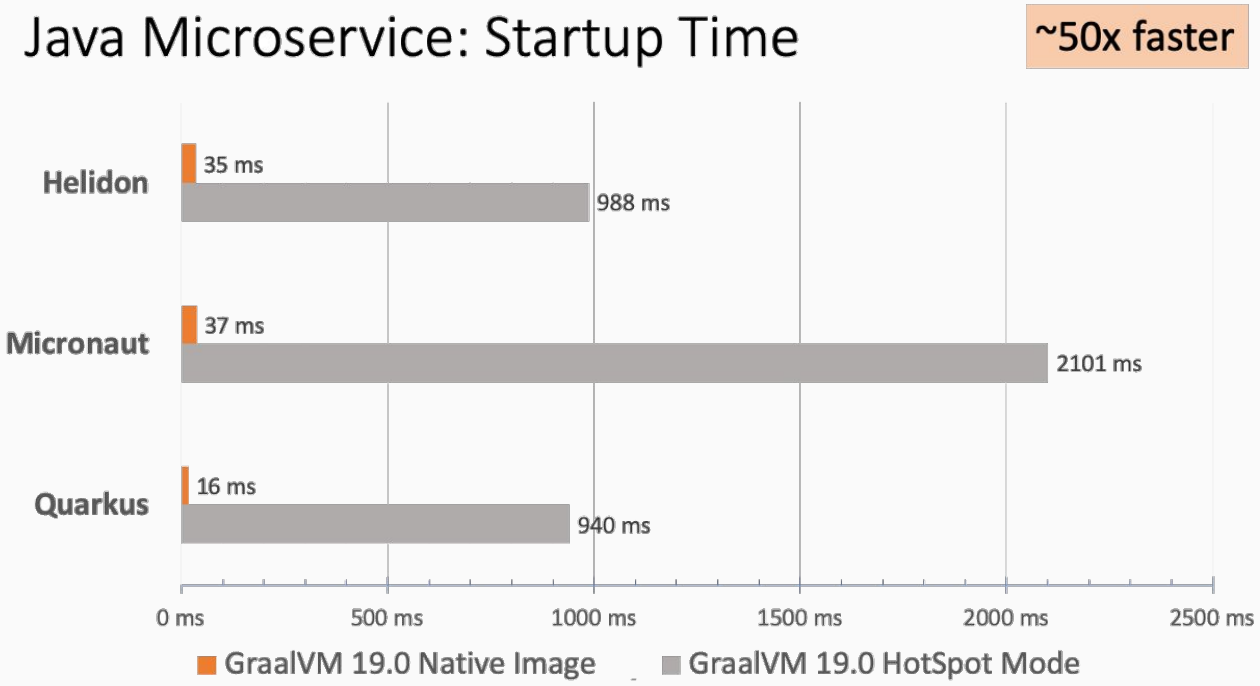
What are the **results** of using **Native Image**?

What are the **results** of using **Native Image**?



Source: https://www.graalvm.org/docs/why-graal/

What are the **results** of using **Native Image**?



Source: https://www.graalvm.org/docs/why-graal/

When to **start** using Java Module, AOT, JIT or Native Image?

When to **start** using Java Module, JIT or AOT?

New Application

Java Module
- Java 9

Just In Time Compilation
- GraalVM

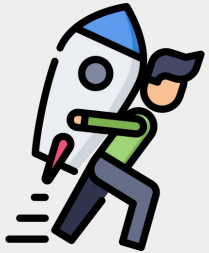
Ahead Of Time Compilation ✅
- Quarkus
- Micronaut

Native Image ✅
- GraalVM
  - ⚠️ Early Adopter Technology

Existent Application

Java Module
- Java 9

Just In Time Compilation ✅
- GraalVM



Java is **dying**?

Thanks a million!

Questions?

/larchanjo

/luram-archanjo