MICRONAUT™

The secret behind of them

Luram Archanjo

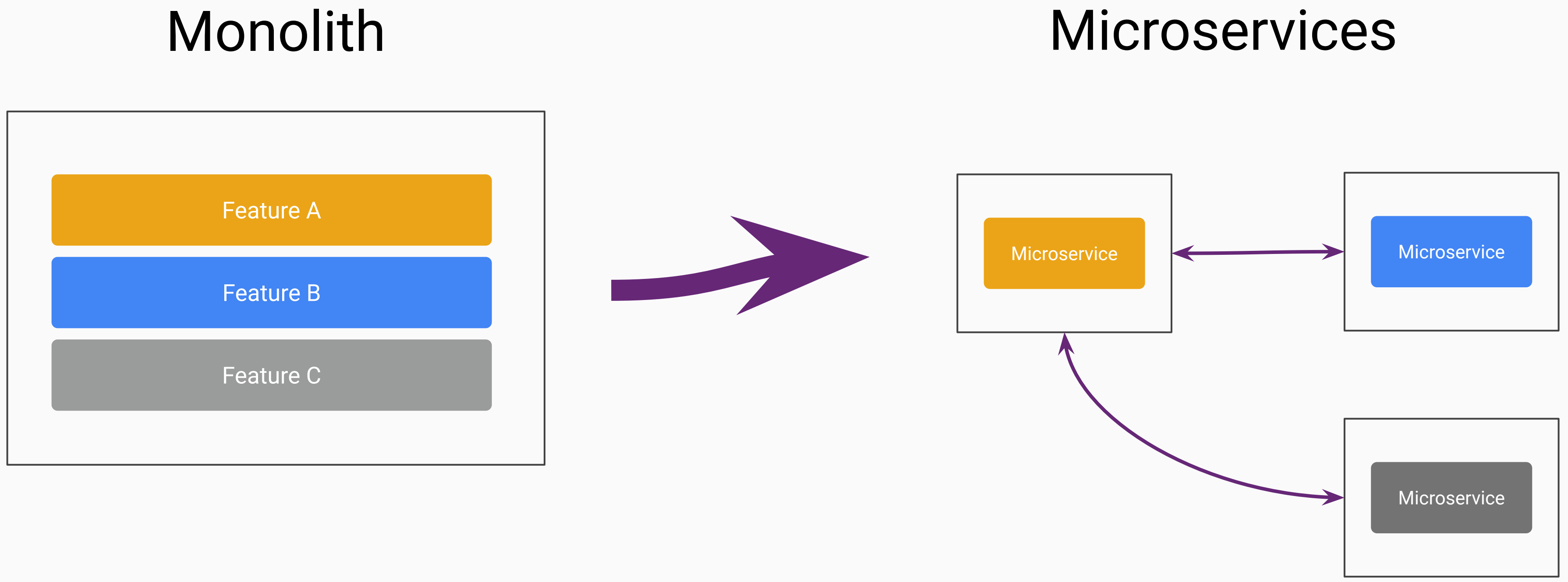
Who am I?

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

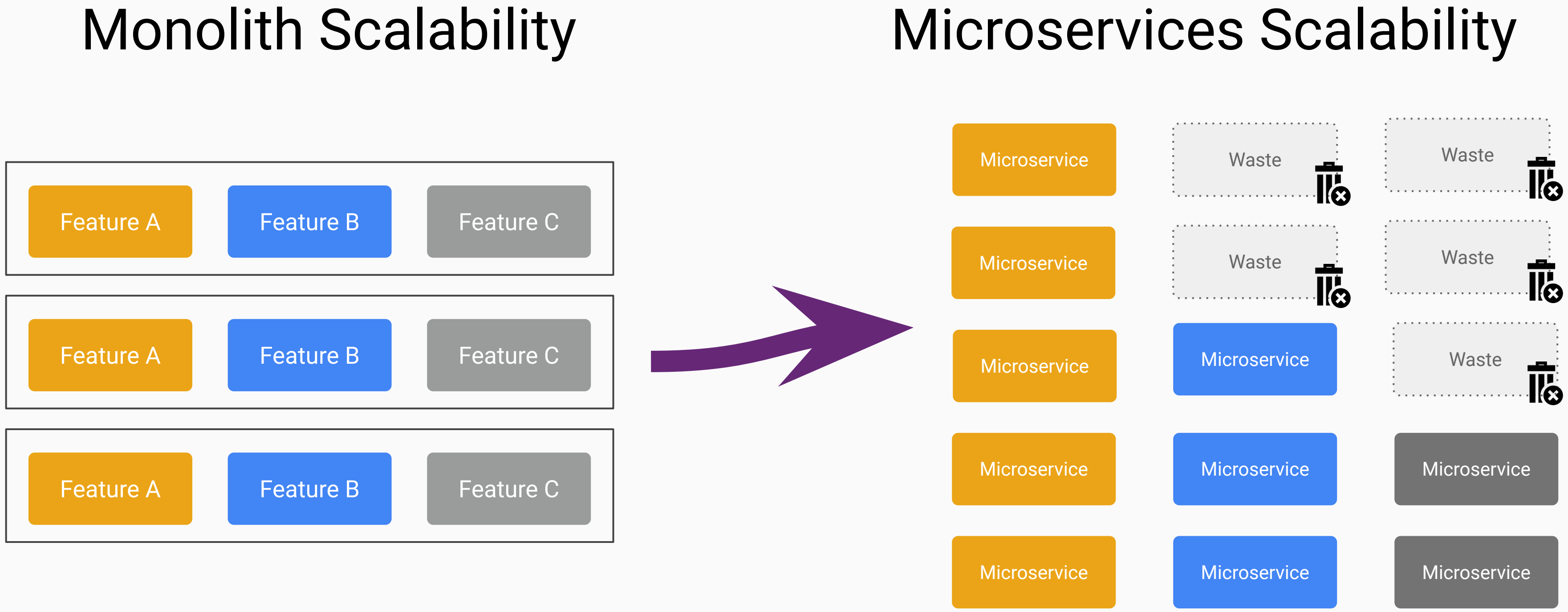
Agenda

- Microservices
- Java & Frameworks
- Ahead of Time (AOT) Compilation
- GraalVM
- Questions

Moving to Microservices



Scalability



Our resources are **finite**!

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

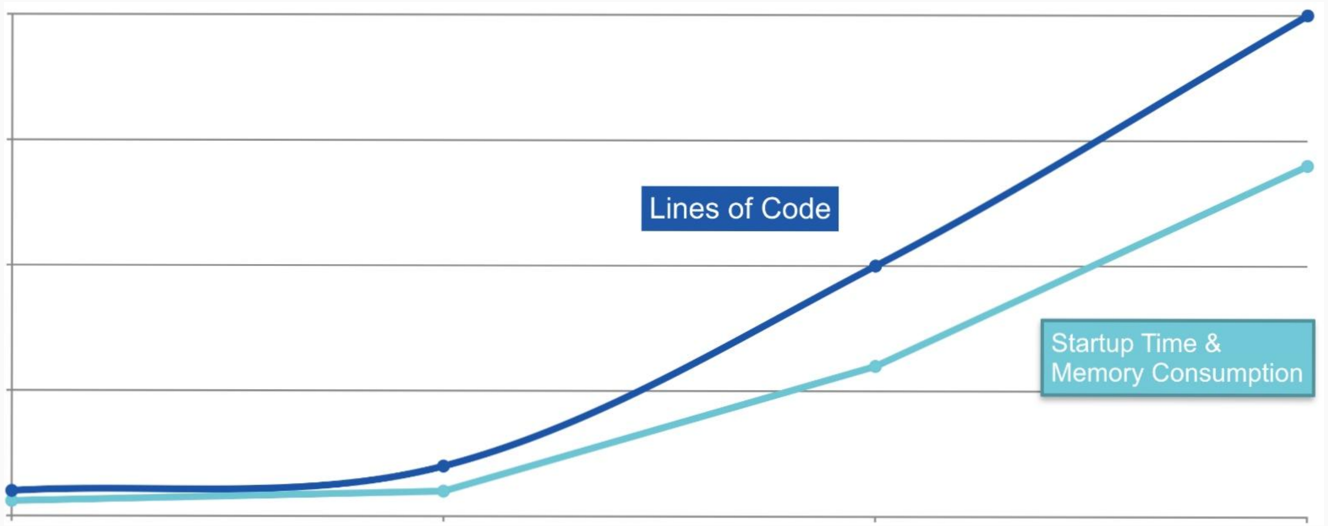
Our frameworks are design to **low memory footprint**?

No, because we've tried to adapt existing **legacy** technologies for **Microservices**

What do Spring and Jakarta EE undertaking? What are the results about it?

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.



The rise of Java **Microframeworks**

Microframeworks

A microframework is a term used to refer to minimalistic web application frameworks:

- Without authentication and authorization
- Without database abstraction via an object-relational mapping.
- Without input validation and input sanitation.

javalin

MICRONAUT

helidon.io

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

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

?

Android

Google Dagger 2

MICRONAUT™

use Ahead of Time (**AOT**) Compilation

What are the **results** of using Ahead of Time (**AOT**) Compilation?

The results of using Ahead of Time (AOT) Compilation

- Startup time around a **second**.
- All Dependency Injection, AOP and Proxy generation happens at **compile time**.
- Can be run with as little as **15mb** Max Heap.

MICRONAUT

QUARKUS

I don't believe, **show me**!

Is it possible **to improve** more?

Yes, with GraalVM™

GraalVM™

GraalVM is an universal virtual machine:

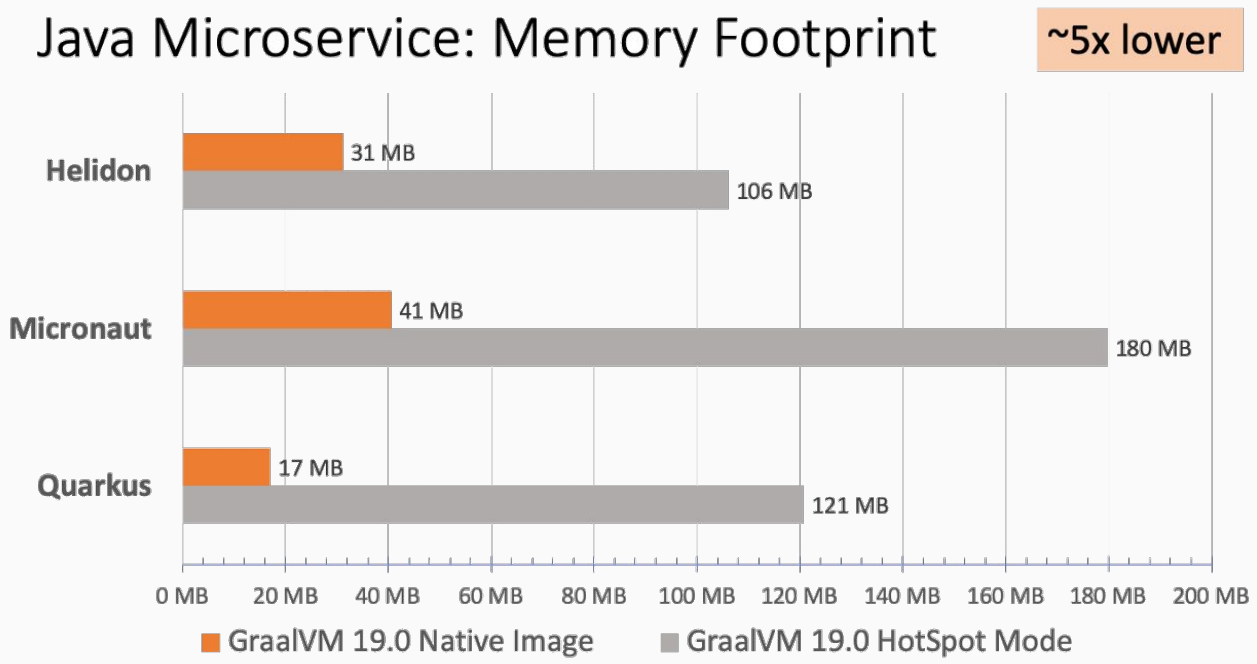
- Runs Java, Scala, Kotlin etc.
- Native image

Native image works well when:

- Little or no runtime reflection is used.
    - Use third party libraries selectively.
- Limited or no dynamic classloading.

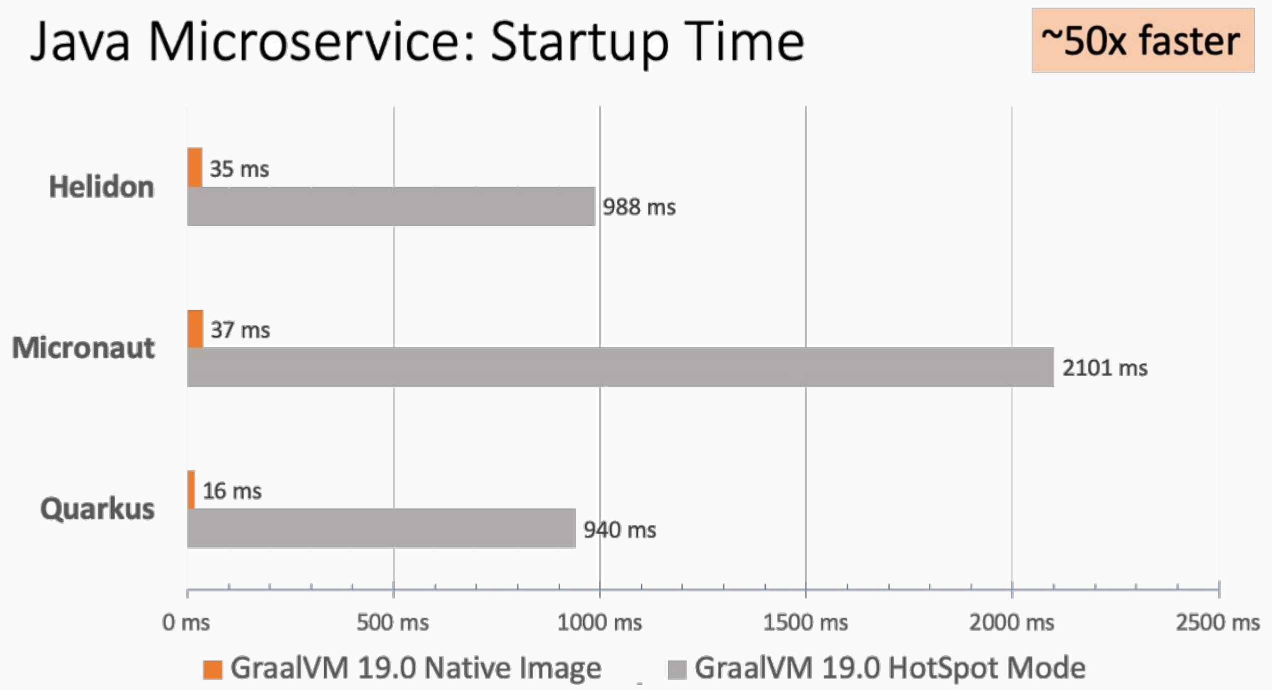
What are the **results** of using **Native Image**?

The results of using Native Image



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

The results of using Native Image



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

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

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

What **else** Micronaut & Quarkus **do**?

There were born in
Microservices and **Cloud** era

There were born in **Microservices** and **Cloud** era

- Observability
  - Open Tracing
    - Zipkin
    - Jaeger
  - Health Checks
  - Metrics

- Fault Tolerance
  - Timeout
  - Retry
  - Circuit Breaker
  - Fallback

- Dependency Injection and Inversion of Control (IoC)

- Blocking or Non-Blocking HTTP Server

They are providing **Java Serverless Application** Adoption

Summary

2º Place

Native Image

- Low memory footprint 5x lower
- Fast Startup 50x lower

1º Place

Ahead of Time (AOT) Compilation

- Low memory footprint
- Fast Startup
- IoC

3º Place

Cloud Native Features

- Observability
- Fault Tolerance
- Distributed Configuration

Which one is the **best**?

You **decide**!

The important thing is that they are **changing** the **Java World**

Thanks a million!

Questions?

/larchanjo

/luram-archanjo