Kotlin on CRaC

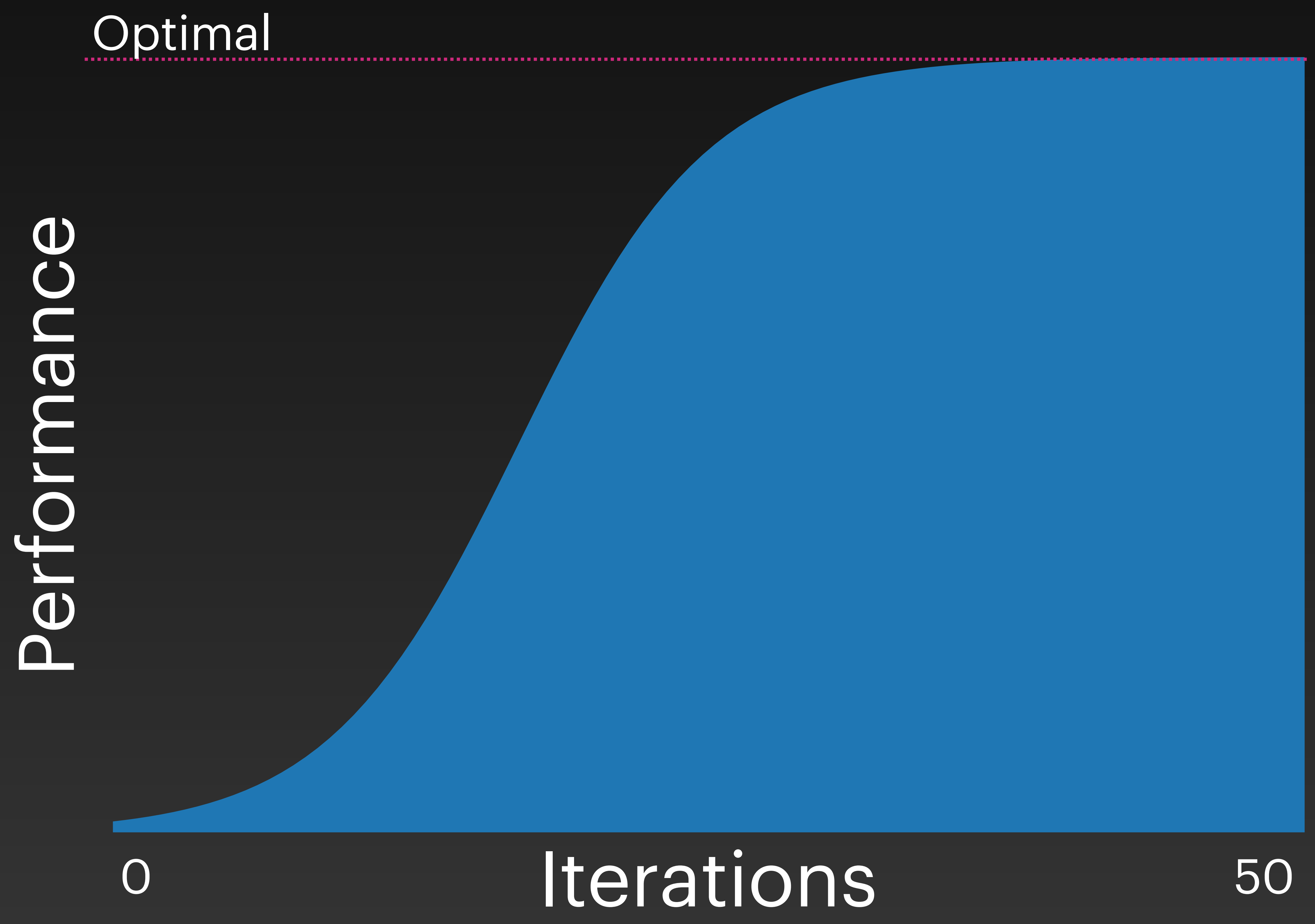
New JVM Features to Speed Things Up

Marc Reichelt

@mreichelt

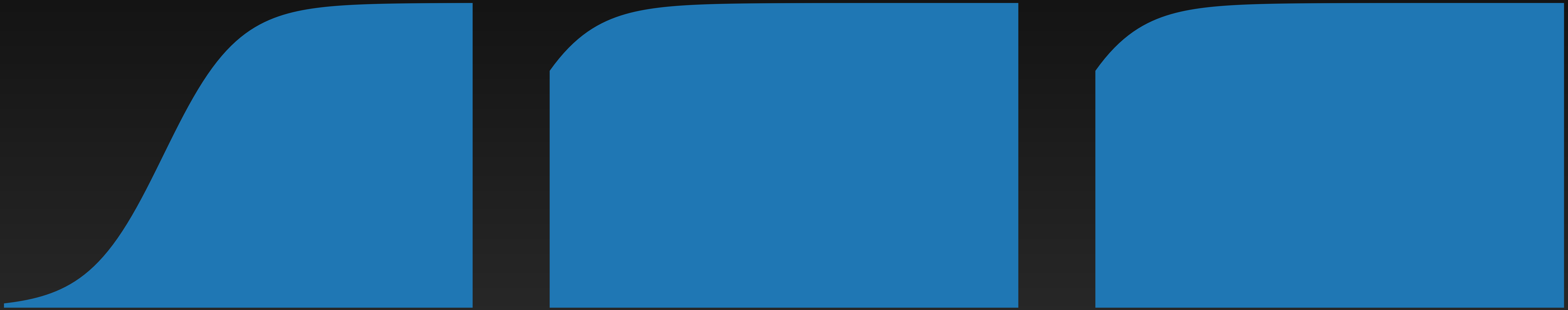
Senior Android Engineer @ Snapp Mobile

Speed of a JVM Program



What if?

What if?



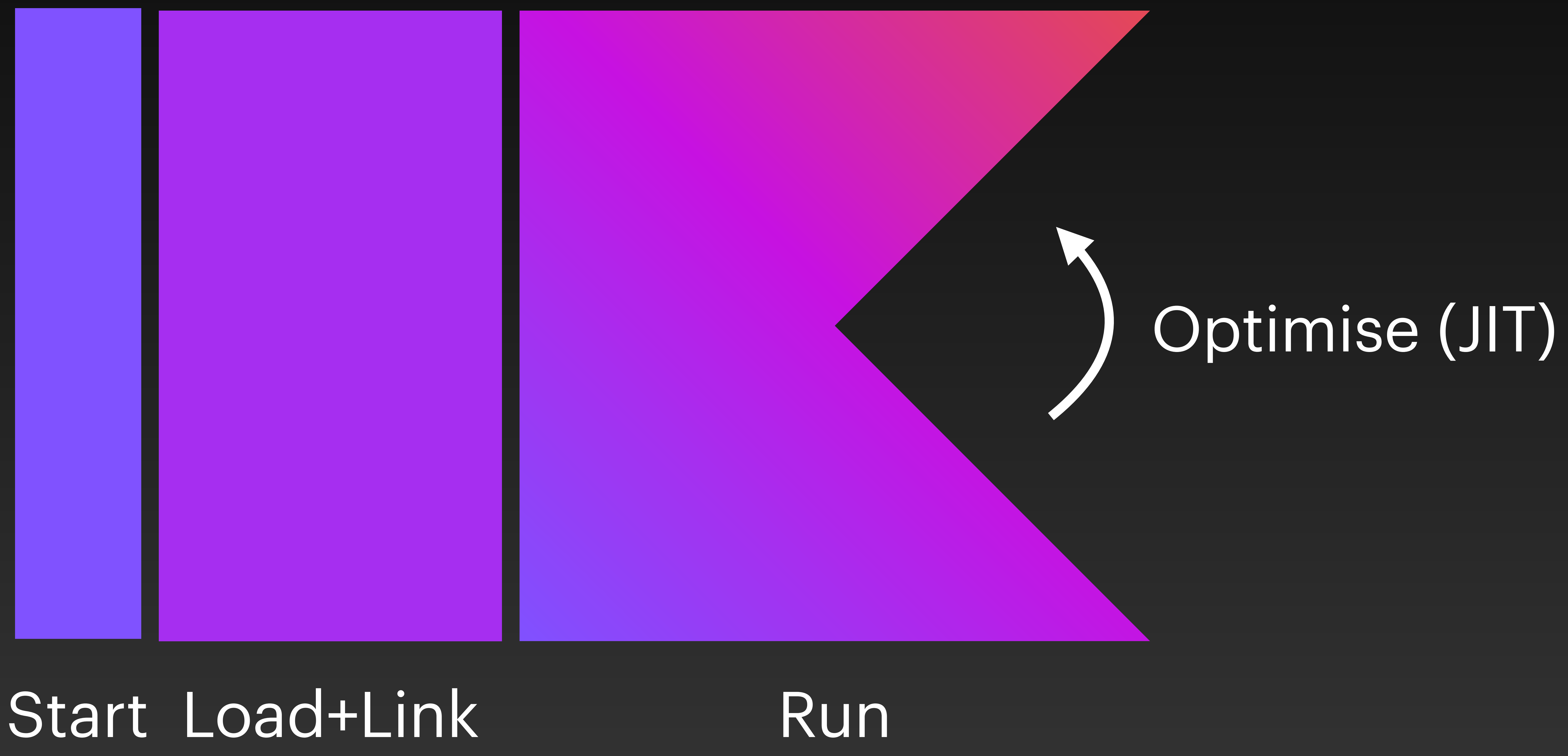
What about GraalVM, or Kotlin Native?

Fast startup, fast runtime, and memory-saving

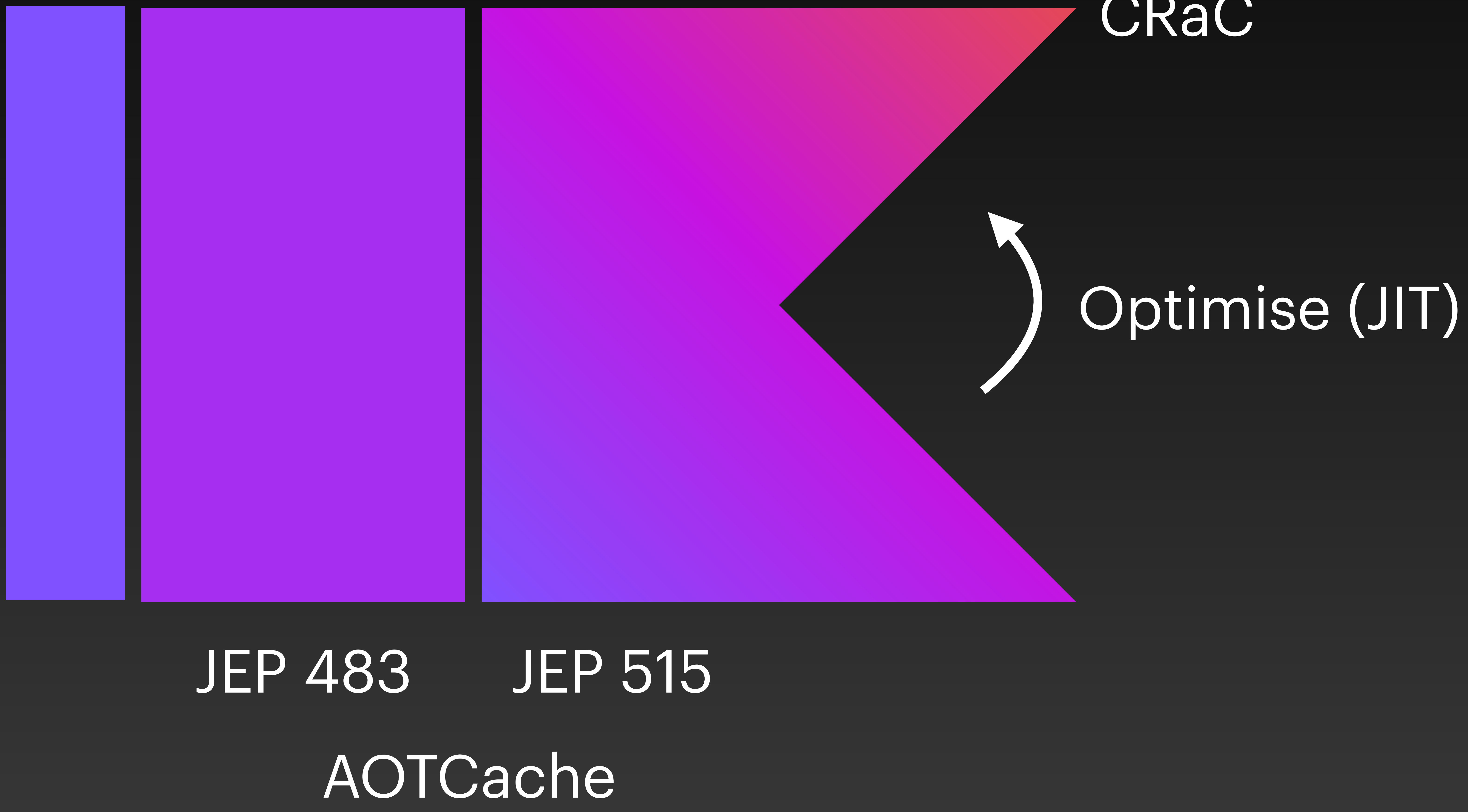
Drawback: long compilation times

No dynamic class loading

Life of a JVM Program



Life of a JVM Program

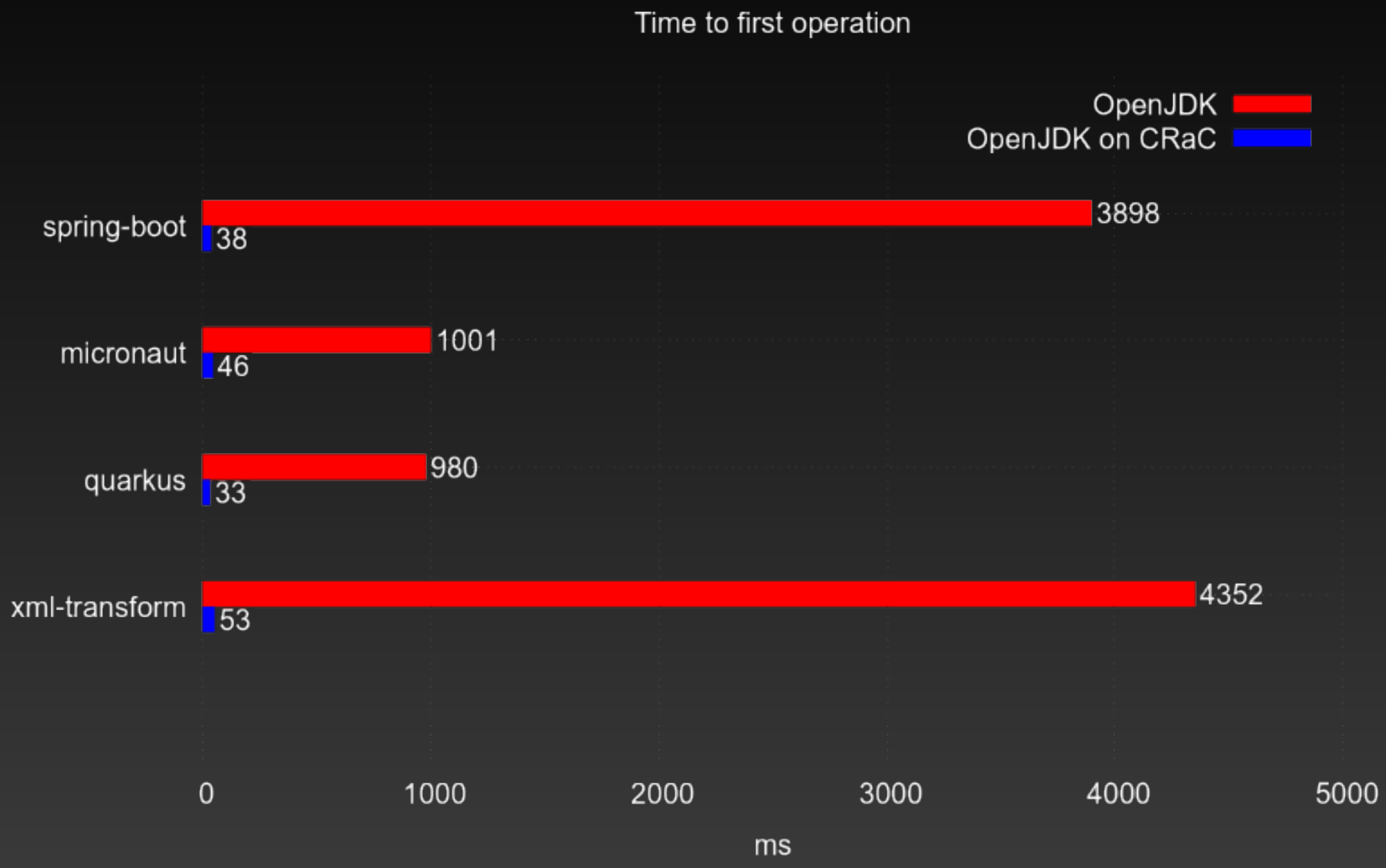


JEP 483	Ahead-of-Time Class Loading & Linking	(JDK 24)
JEP 515	Ahead-of-Time Method Profiling	(JDK 25)
CRaC	Coordinated Restore at Checkpoint*	

JDK with CRaC support, e.g. Azul Zulu JDK

Linux-only for now

CRaC Performance



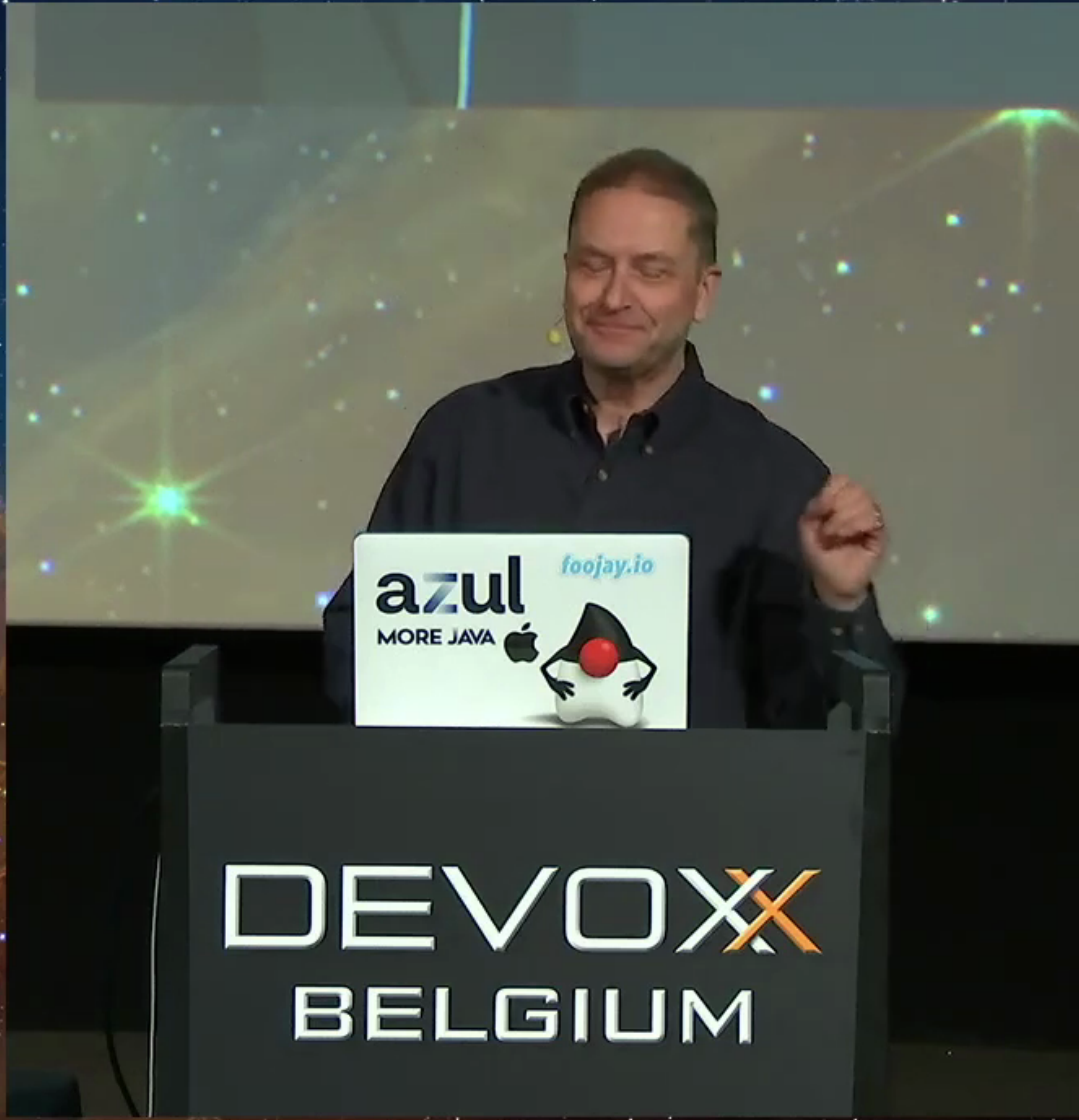
from github.com/CRaC/docs

Talk Recommendation

azul

Java on CRaC

Presented by Simon Ritter, Deputy CTO | Azul



The Challenge™

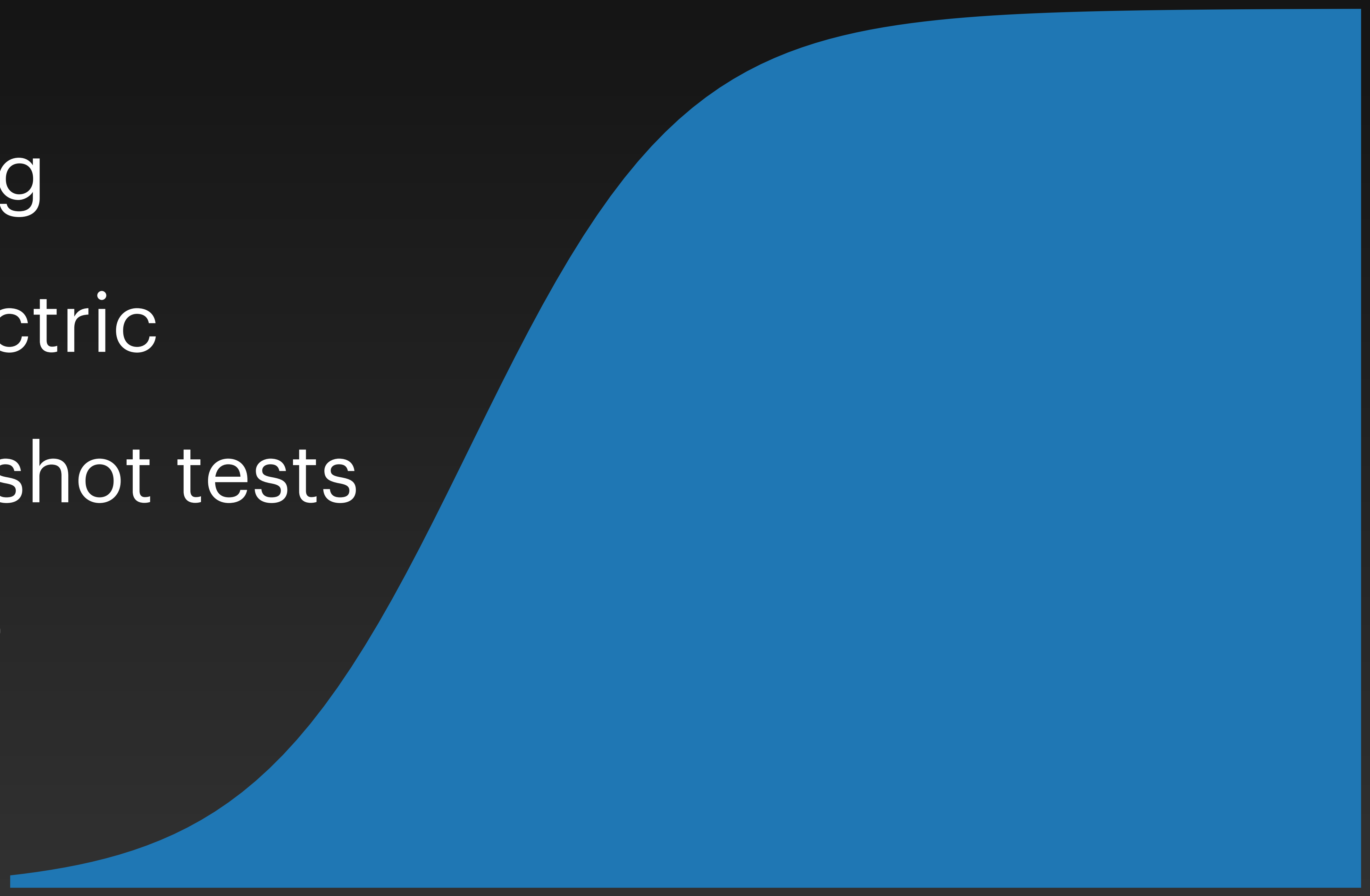
Can we make unit tests with Gradle faster?

Mocking

Robolectric

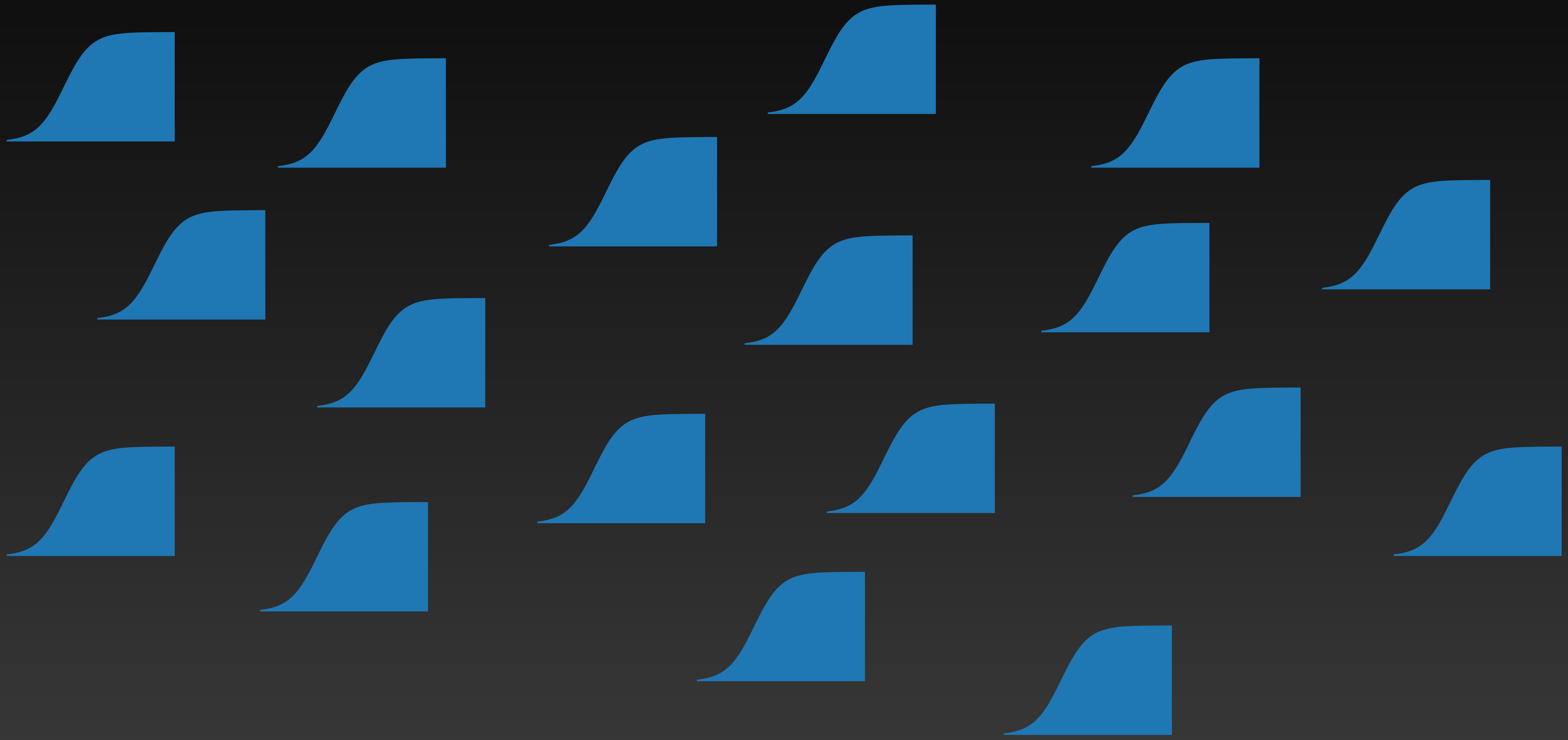
Screenshot tests

Your lib



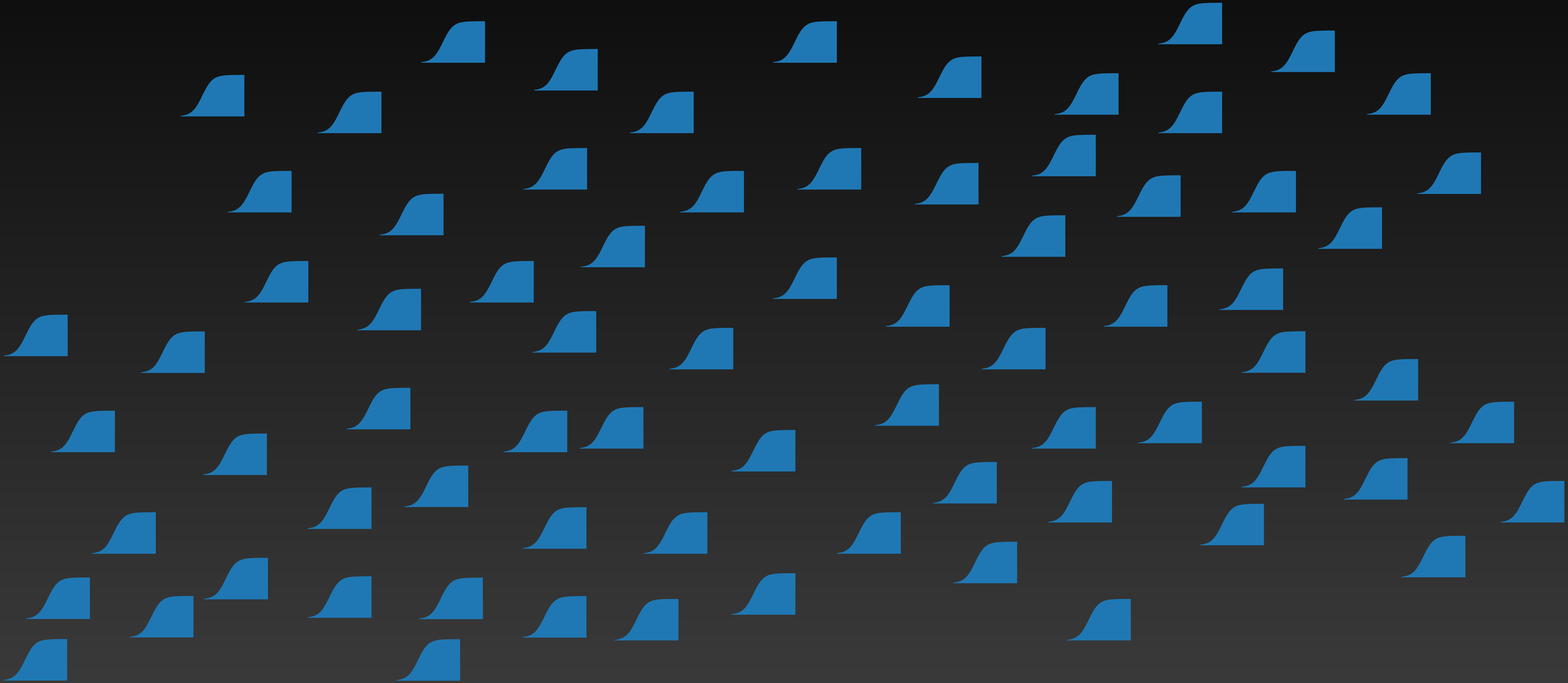
The Challenge™

Can we make unit tests with Gradle faster?

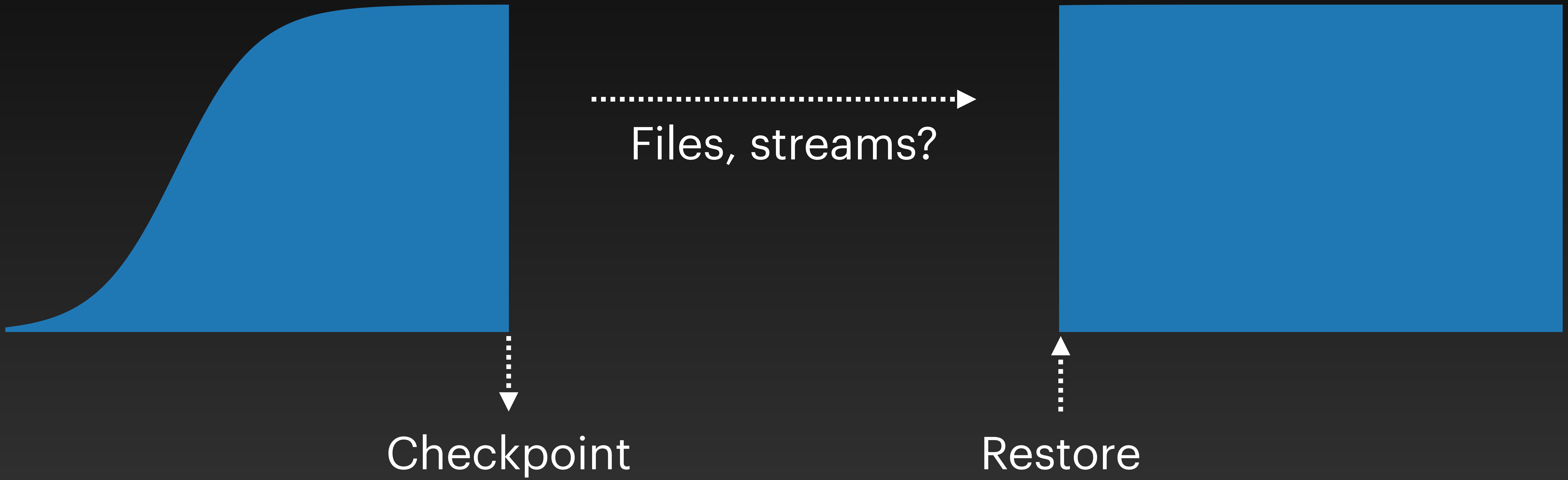


The Challenge™

Can we make unit tests with Gradle faster?



CRaC



Files, streams?

Checkpoint

Restore

📁 crac-files

```kotlin
// mvn dependency org.crac:crac


class ExampleResource : Resource {
    override fun beforeCheckpoint(context: Context<out Resource>) {
        // close files & streams
    }

    override fun afterRestore(context: Context<out Resource>) {
        // reopen files & streams
    }
}


// then in your code:
Core.getGlobalContext().register(ExampleResource())
```

```
// mvn dependency org.crac:crac


class ExampleResource : Resource {
    override fun beforeCheckpoint(context: Context<out Resource>) {
        // close files & streams
    }

    override fun afterRestore(context: Context<out Resource>) {
        // reopen files & streams
    }
}


// then in your code:
Core.getGlobalContext().register(ExampleResource())
```

```
// mvn dependency org.crac:crac


class ExampleResource : Resource {
    override fun beforeCheckpoint(context: Context<out Resource>) {
        // close files & streams
    }

    override fun afterRestore(context: Context<out Resource>) {
        // reopen files & streams
    }
}


// then in your code:
Core.getGlobalContext().register(ExampleResource())
```

```kotlin
// mvn dependency org.crac:crac


class ExampleResource : Resource {
    override fun beforeCheckpoint(context: Context<out Resource>) {
        // close files & streams
    }

    override fun afterRestore(context: Context<out Resource>) {
        // reopen files & streams
    }
}


// then in your code:
Core.getGlobalContext().register(ExampleResource())
```

```kotlin
// mvn dependency org.crac:crac


class ExampleResource : Resource {
    override fun beforeCheckpoint(context: Context<out Resource>) {
        // close files & streams
    }

    override fun afterRestore(context: Context<out Resource>) {
        // reopen files & streams
    }
}


// then in your code:
Core.getGlobalContext().register(ExampleResource())
```

```kotlin
// mvn dependency org.crac:crac


class ExampleResource : Resource {
    override fun beforeCheckpoint(context: Context<out Resource>) {
        // close files & streams
    }

    override fun afterRestore(context: Context<out Resource>) {
        // reopen files & streams
    }
}


// then in your code:
Core.getGlobalContext().register(ExampleResource())
```

Let’s go live

✨ The Future ✨

JEP 516: Ahead-of-Time Object Caching with Any GC

JEP draft: Ahead-of-Time Code Compilation

Project Leyden

My wish: AOTCache more robust to class path changes

Enjoy the Conference!

Marc Reichelt

@mreichelt

github.com/mreichelt/kotlin-on-crac

Senior Android Engineer @ Snapp Mobile