
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

BOA306

Designing and delivering cloud-optimized Java applications

Mohammed Fazalullah

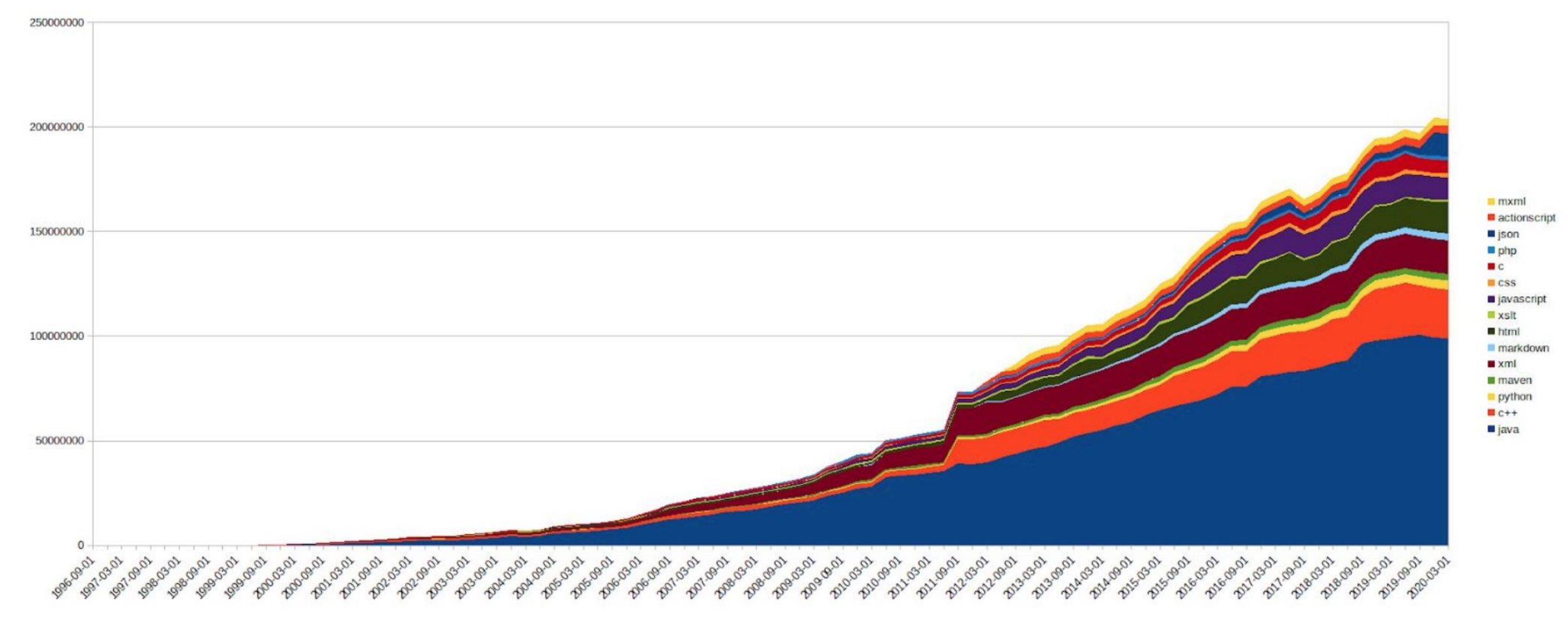
Senior Developer Advocate
Amazon Web Services

Viktor Vedmich

Senior Developer Advocate
Amazon Web Services

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Java: 25 years and going



Apache Software Lines of code 1996-2020. Source: [The Apache Software Foundation – Annual Report – FY2020]

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Java ecosystem overview

Spring journey to AWS Lambda

Accelerating with modern frameworks

Other considerations

Summary

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Motivation

Java plays an essential role when building reliable software architectures

Spring Boot is the most popular Java application framework with more than 60%* of Java developers reportedly using it

Broad ecosystem (libraries, tools, and resources) and developer experience

Over the past years companies invested in these technologies

New feature development: Spring 3.0 with GraalVM support, Project Leyden, CRaC, Java release cycle

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

* Source: https://www.jrebel.com/blog/2021-java-technology-report

Challenges with Java and Spring in serverless

Java initially was designed around factors that differ from modern cloud-native requirements

There is a paradigm shift from always-on serverful applications to short-lived serverless functions

Large enterprise frameworks often perform expensive operations during initial JVM startup time

But why is this a problem?

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Spring Boot across compute

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

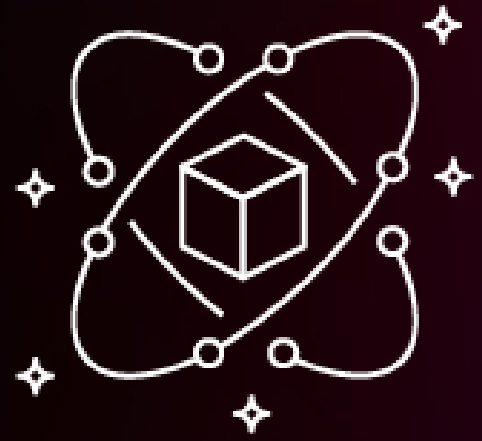
A Spring journey to AWS Lambda



1. Migrate



2. Optimize



3. Refactor



4. Accelerate

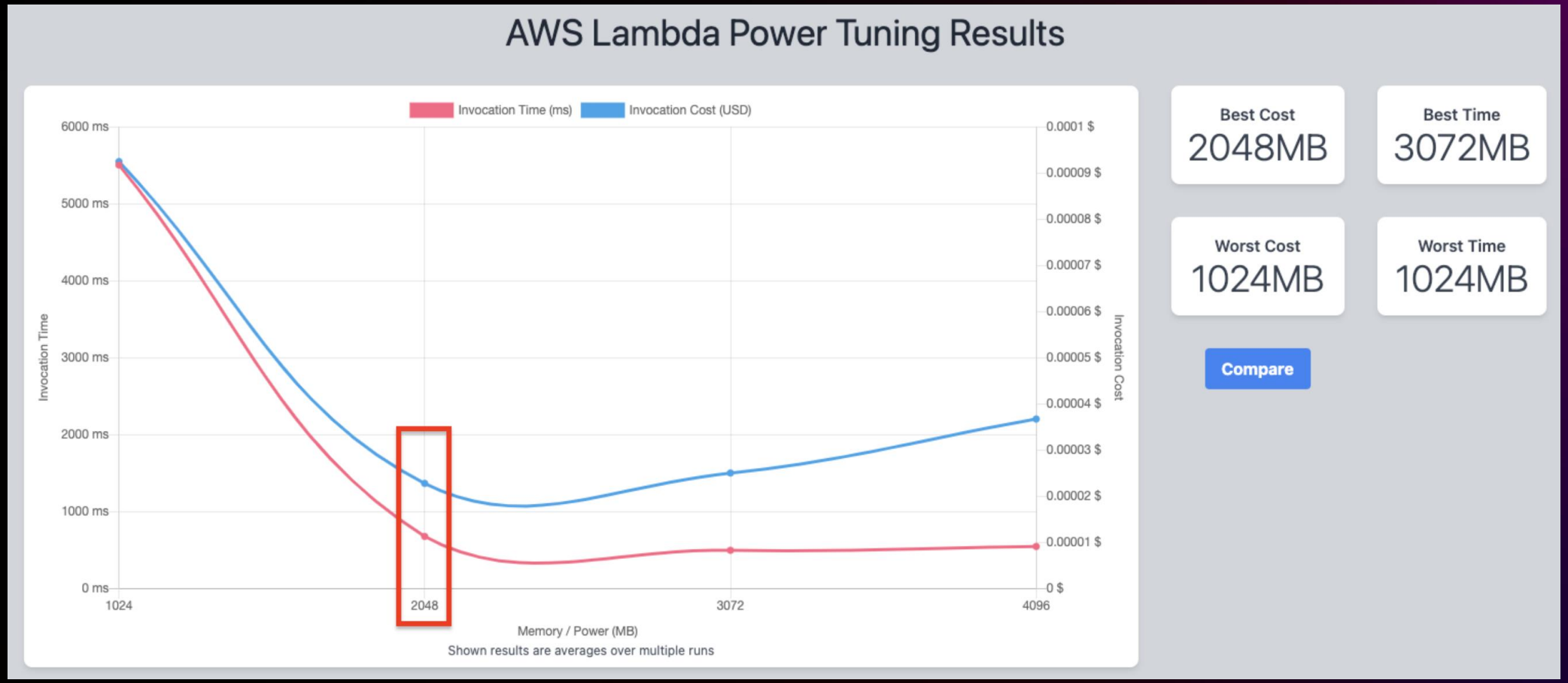
aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Optimize

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Rightsize memory – AWS Lambda Power Tuning



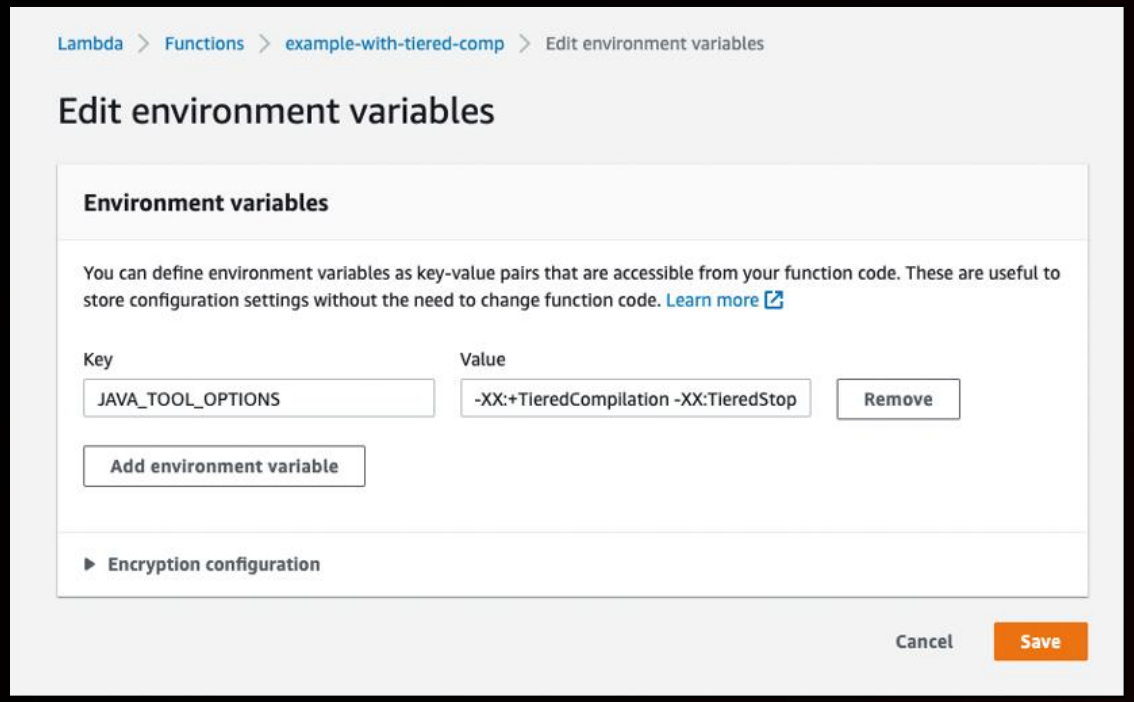
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Source: https://github.com/alexcasalboni/aws-lambda-power-tuning

Tiered compilation

Use optimized C1 compiler for faster application start up

Add an environment variable to change the compiler level



	Init duration	
Without tiered	11.378	
With tiered	6.725	- 40%

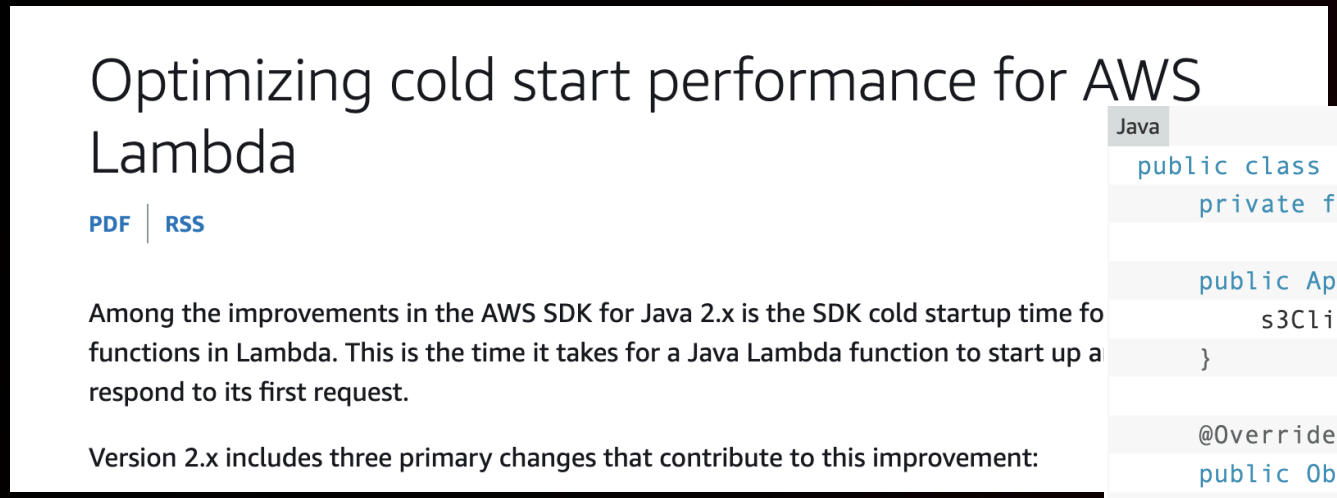
Results will vary based on your application

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Follow general AWS Java best practices

Use the new AWS SDK version (v2) with proper configuration and optimized HTTP clients

Initialize expensive operations outside the handler code



Optimizing cold start performance for AWS Lambda

PDF | RSS

Among the improvements in the AWS SDK for Java 2.x is the SDK cold startup time fo
functions in Lambda. This is the time it takes for a Java Lambda function to start up a
respond to its first request.

Version 2.x includes three primary changes that contribute to this improvement:

Java

```java
public class App implements RequestHandler<Object, Object> {
    private final S3Client s3Client;

    public App() {
        s3Client = DependencyFactory.s3Client();
    }

    @Override
    public Object handleRequest(final Object input, final Context contex
         s3Client.listBuckets();
        // Function logic here
    }
}
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Refactor

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using lightweight dependencies

- Trade-off between feature richness and performance
  - spring-data-jpa vs. spring-data-jdbc
  - jackson vs. jackon-jr, Log4J2 vs. SLF4J SimpleLogger

	Duration	
Spring Data JPA	6.725	
Spring Data JDBC	5.052	- 25%

Results will vary based on your application #YMMV

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Accelerate

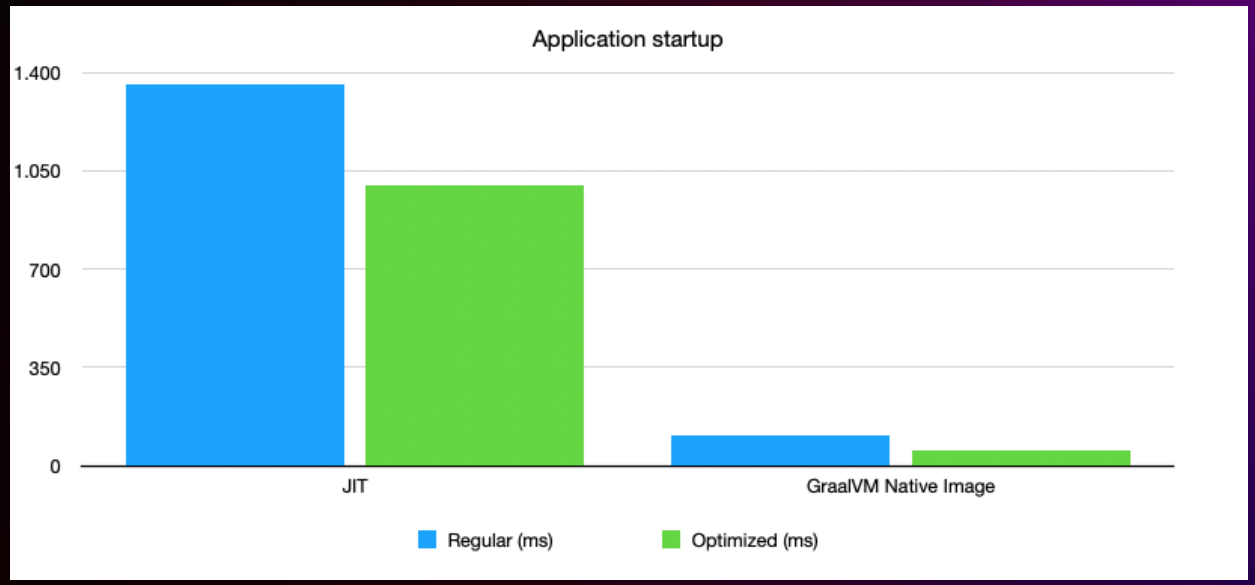
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Compile your application ahead of time (AOT)

GraalVM creates a native image of your application at build time

Frameworks like Quarkus and Micronaut integrate GraalVM and add further benefits such as build-time dependency injection

Some JVM features are not supported



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Other considerations

Function granularity and least privilege

Connection pooling

Endpoint-based functionality (Spring Actuator, Metrics scraping)

File system logging

Caching

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Resources

Micronaut
https://micronaut.io/

GraalVM
https://www.graalvm.org/

Serverless Java Container
https://github.com/awslabs/aws-serverless-java-container

AWS Lambda Power Tuning
https://docs.aws.amazon.com/lambda/latest/operatorguide/profile-functions.html

SVS310: From Serverful to Serverless Java with AWS Lambda
https://serverlessland.com/reinvent2022/svs310

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.