Reducing Latencies

Amazon Corretto and Micronaut™

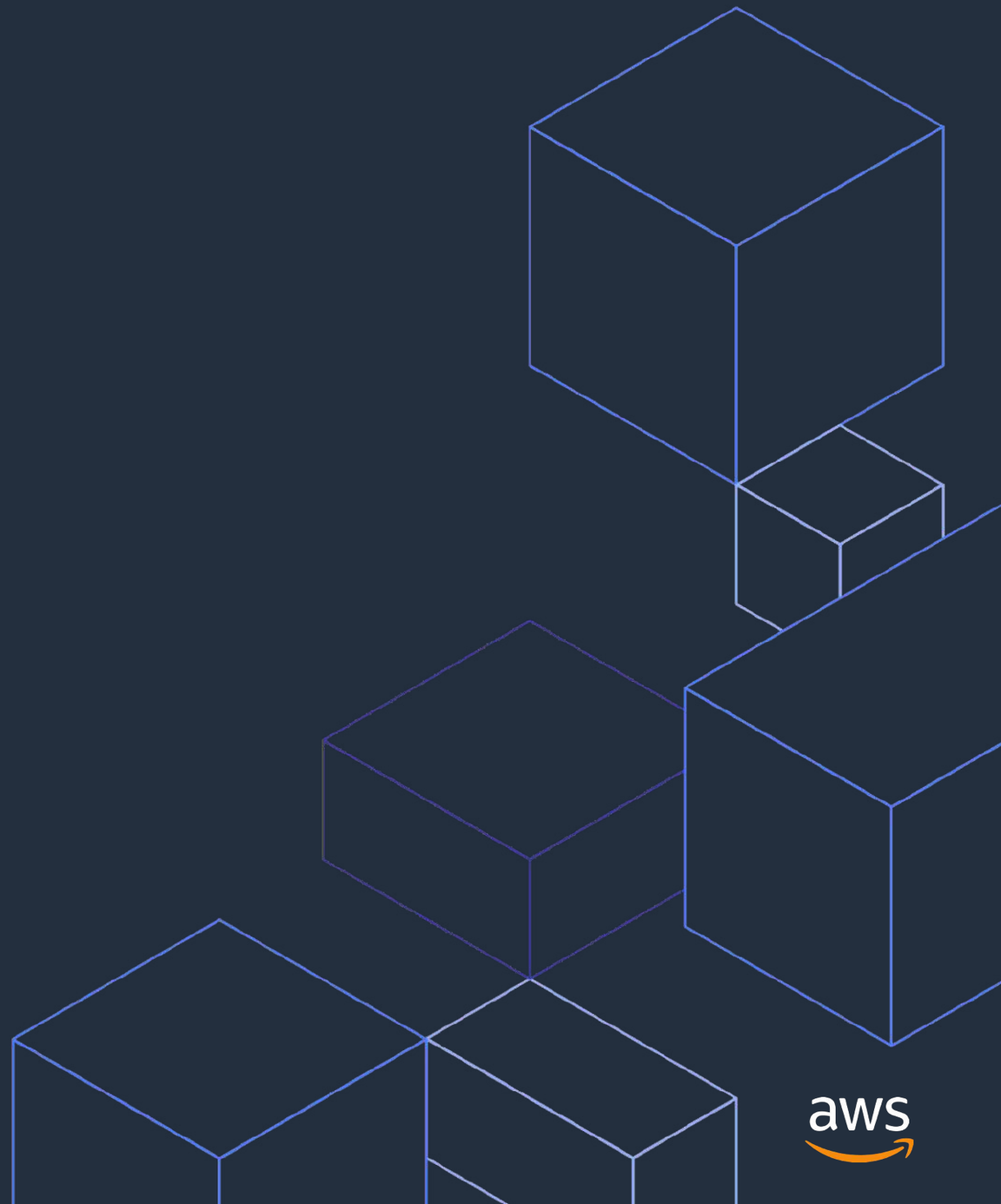
Alvaro Sanchez-Mariscal – SSE at Object Computing
Azeem Jiva – SDM Corretto JVM
10/30/2020

Table of contents

- Corretto
- Shenandoah
- GC Latencies
- Micronaut Overview
- Demo
- Links to Resources
- Q&A

aws

Corretto Overview



Amazon Corretto

Downstream distribution of OpenJDK

No-cost long-term support

At least quarterly releases

Drop-in replacement for Oracle JDK

Multiplatform: Linux, Windows, macOS, Docker

TCK certified

https://aws.amazon.com/corretto/

aws

Why Corretto?

Customer obsession

Ability to quickly fix bugs

Performance improvements for our use cases

What enabled Corretto adoption in Amazon?

Drop-in compatibility (barring non-OpenJDK features)

Automated deployment pipelines

Cross-company profile visibility

Upside: performance goodies, backports, bug fixes

aws

Why external binary of Corretto?

OpenJDK on Amazon Linux

End of public updates for Oracle JDK 8 in January 2019

Customer obsession

aws

How do we support?

No-cost security patches, quarterly

GitHub and Stackoverflow

AWS Premium support customers can file tickets

How do we collaborate?

Patches go upstream

Working with Red Hat and others on JDK 8, JDK 11, Shenandoah and other projects

aws

Shenandoah Overview

What is Shenandoah?

A mostly concurrent garbage collector, developed by Red Hat.

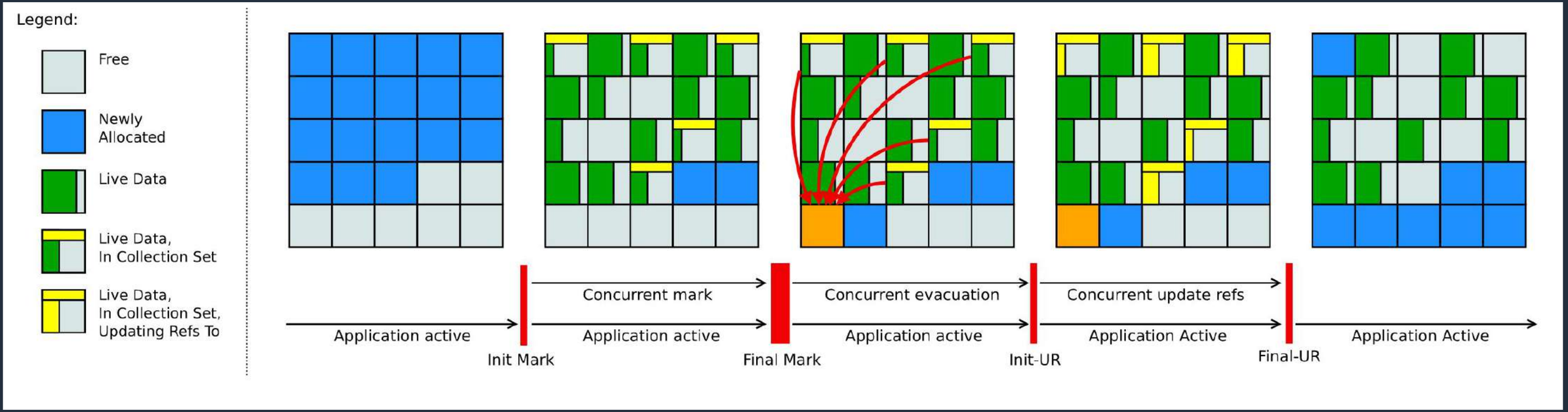
Advantages:

- Pause times are within 0..10ms
- Pause times are not proportional to size of heap

Disadvantage:

- Throughput losses are within 0..15%
- Requires more free heap

How does it work?



- Flip “briefly” suspends threads to initiate GC
- Thereafter, parallel application and GC threads “share” access to heap memory
- Coordination requires synchronization locks and memory fences
- Empirically, reads are 10x more frequent than writes

aws

Shenandoah pauses

Shenandoah's pauses are dominated by a set of root set operations:

- Local variables
- References embedded in generated code
- Interned Strings
- References from classloaders (e.g. static final references)
- JNI, JVMTI references

Larger root set means longer pauses with Shenandoah

Micronaut Overview

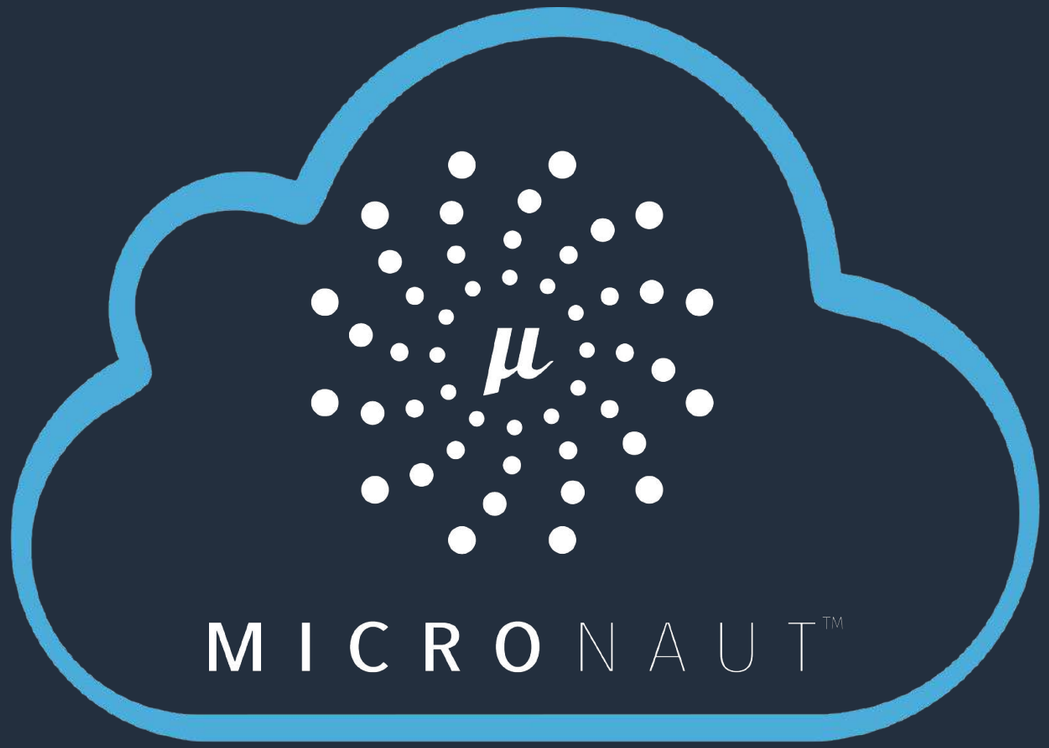
Micronaut Overview

- Microservices and serverless framework for the JVM.
- Natively Cloud Native.
- Ultra-lightweight and reactive.
- AOT compilation.
- Fast startup time, low memory consumption.
- Polyglot: Java, Kotlin and Groovy.

MICRONAUT™

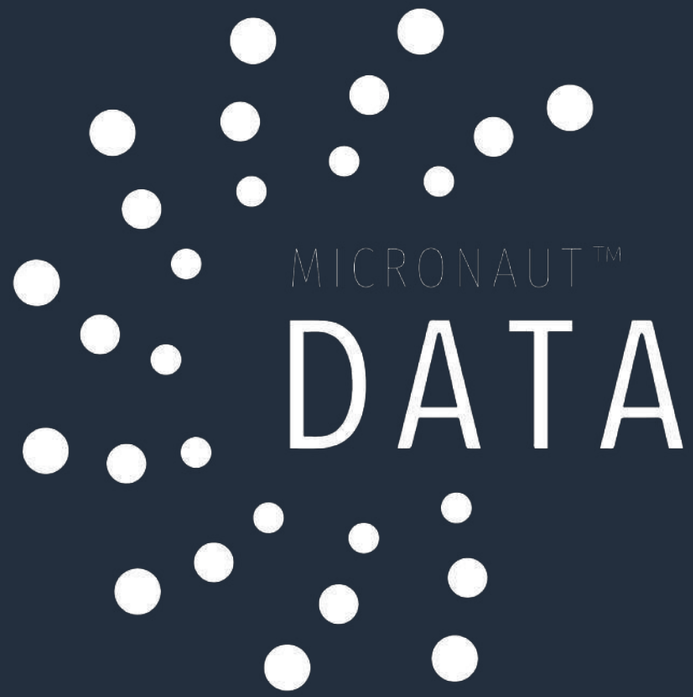
aws

Natively Cloud Native



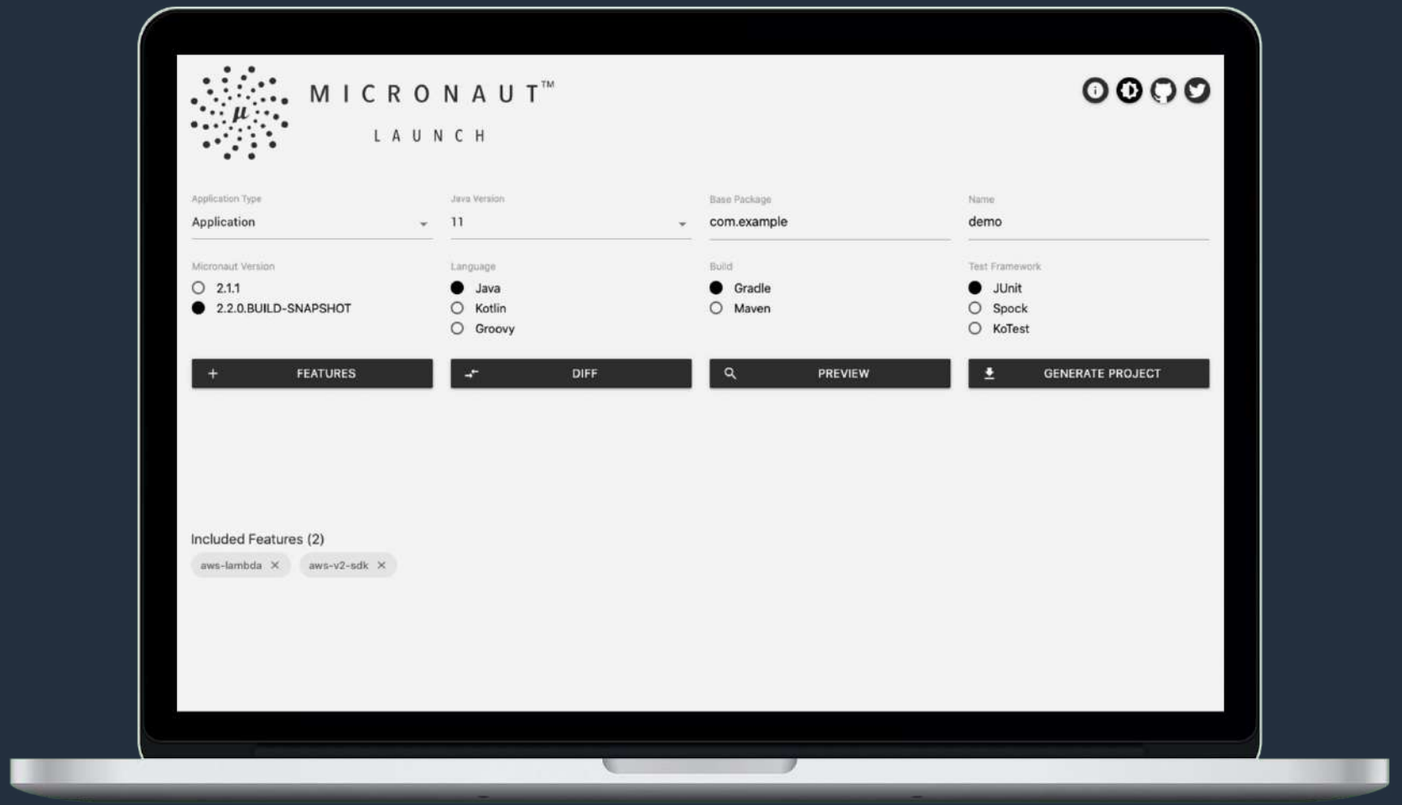
aws

Flexible Data Access



aws

Getting started: Micronaut Launch



https://micronaut.io/launch

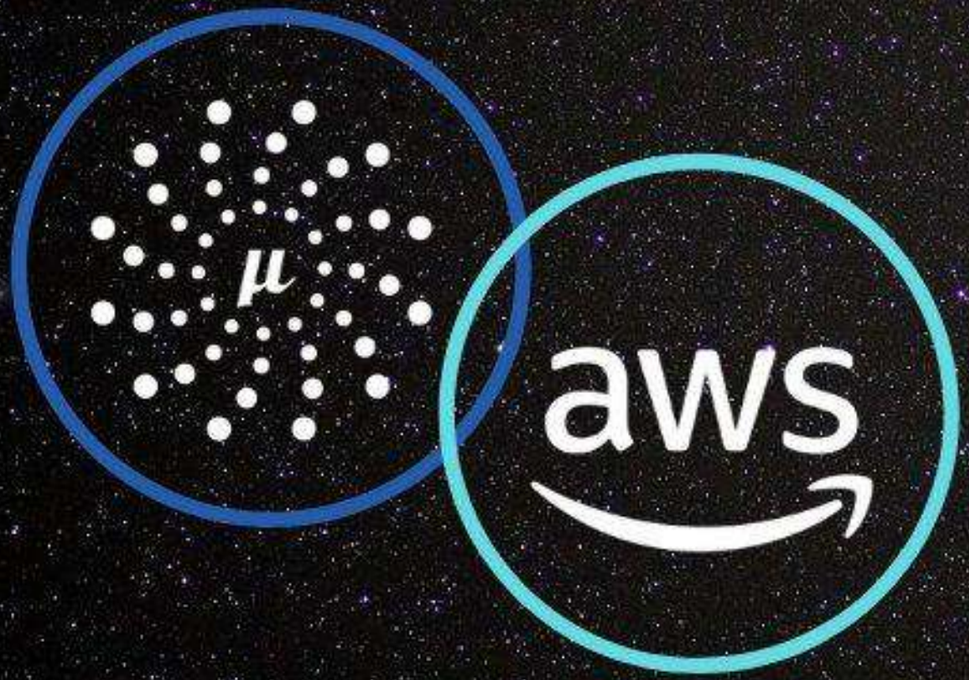
Getting started: Micronaut CLI

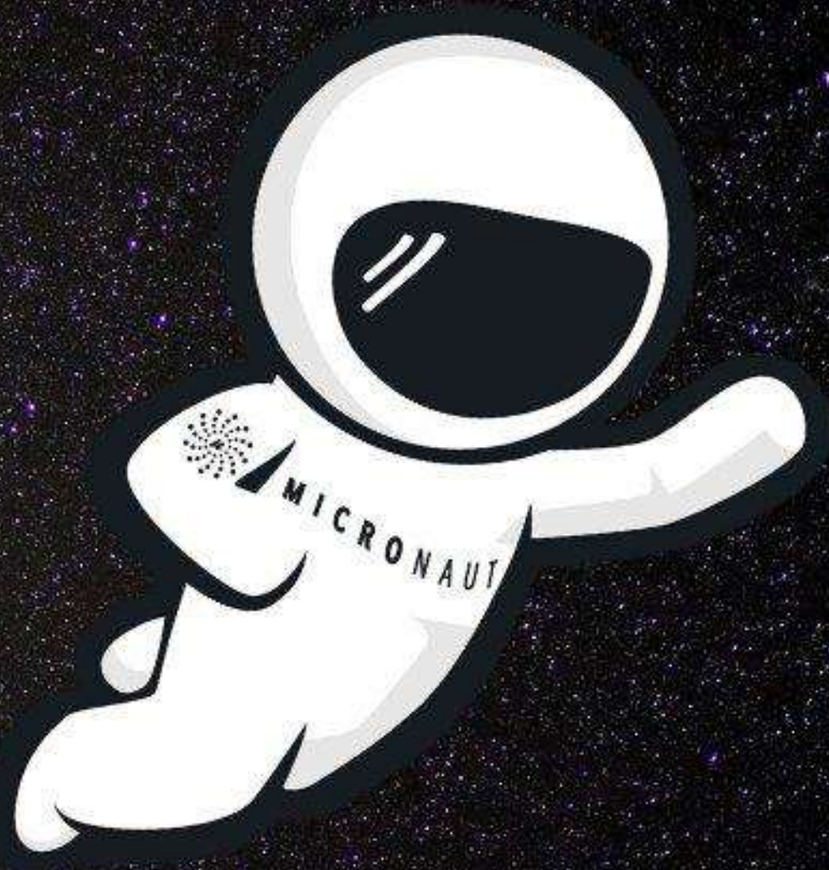
aws

Micronaut ❤ AWS

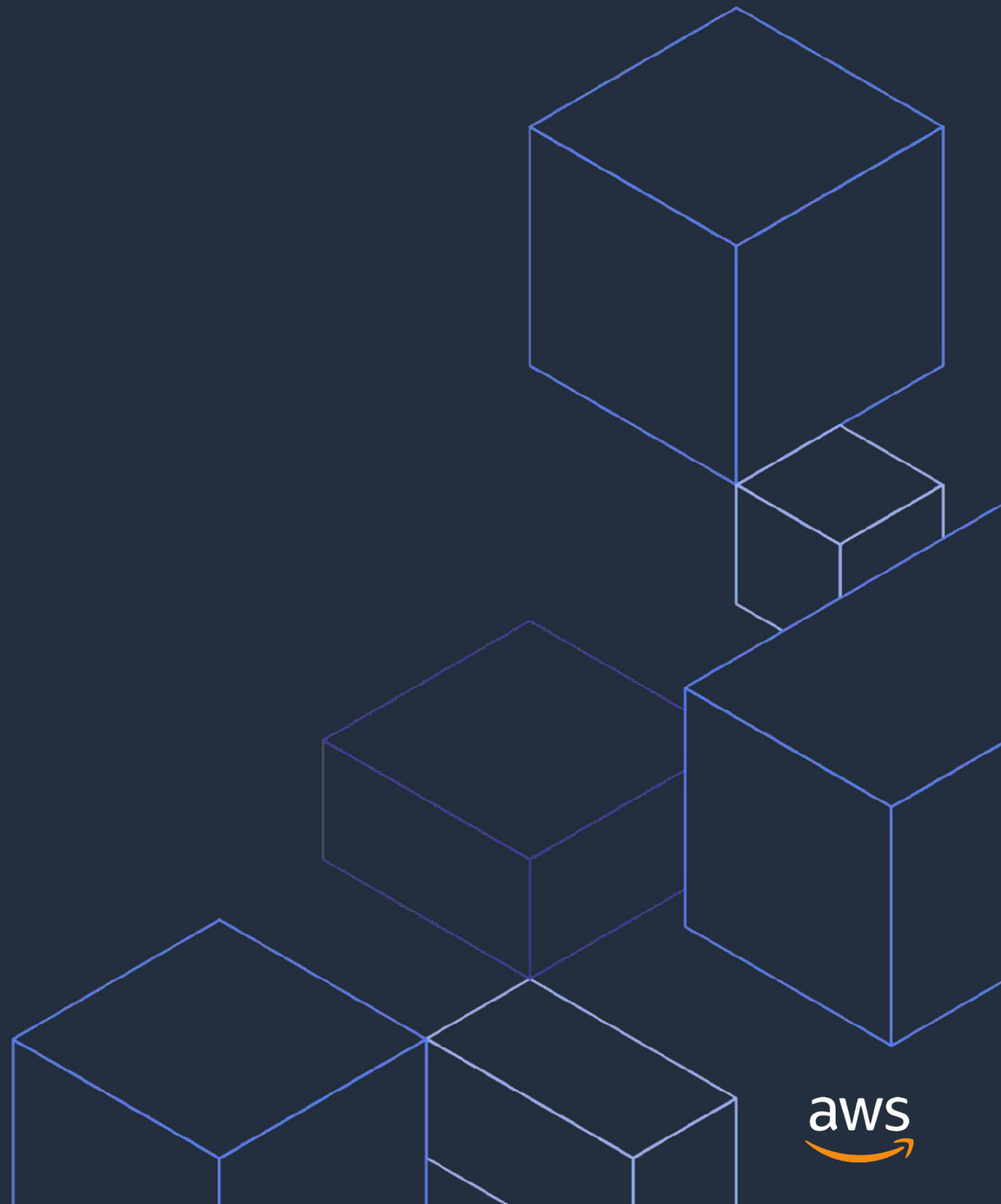
- Write applications or serverless functions.
- AWS SDK v1 and v2 configurations.
- Lambda support.
    - Corretto 8/11 runtimes.
    - Custom runtime (GraalVM).
- Alexa support.
    - Skills, flash briefings, etc.

Micronaut Is Now Certified to Run on Amazon Corretto





Demo



Demo

- Sample application: Micronaut Data JDBC.
    - https://github.com/alvarosanchez/micronaut-corretto-shenandoah

- OpenJDK 11 using G1 vs Corretto 11 using Shenandoah.
- Warmup: 10K req/s using 10 threads and 500 connections during 20 seconds.
- Load test: 50K req/s using 10 threads and 500 connections during 60 seconds.

aws

Versions and modifiers

```
$ java -version
openjdk version "11.0.9" 2020-10-20
OpenJDK Runtime Environment AdoptOpenJDK (build 11.0.9+11)
OpenJDK 64-Bit Server VM AdoptOpenJDK (build 11.0.9+11, mixed mode)

$ java -jar application.jar
```

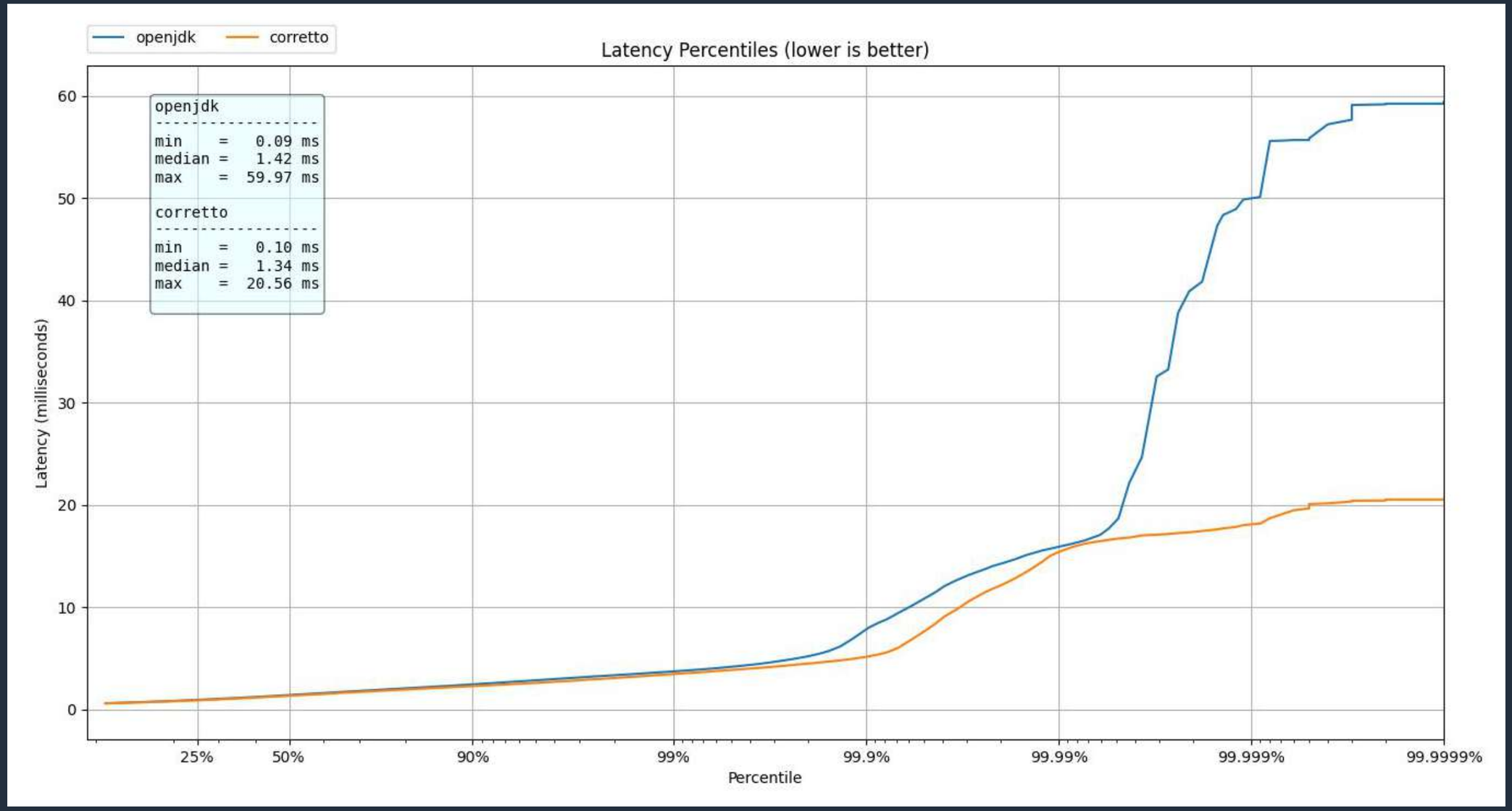

aws

Versions and modifiers

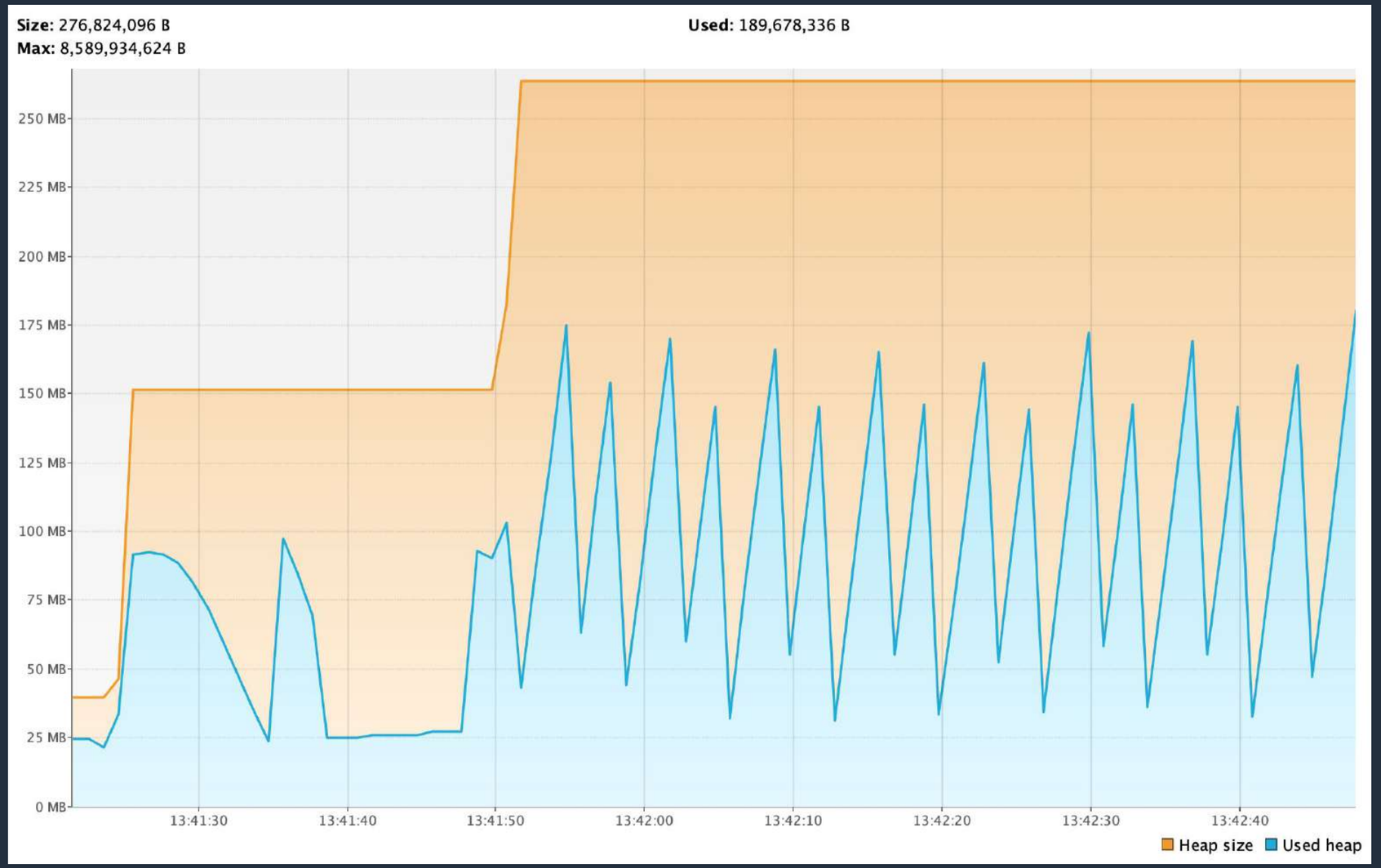
```
$ java -version
openjdk version "11.0.9" 2020-10-20 LTS
OpenJDK Runtime Environment Corretto-11.0.9.11.1 (build 11.0.9+11-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.9.11.1 (build 11.0.9+11-LTS, mixed mode)

$ java -XX:+UnlockExperimentalVMOptions -XX:+UseShenandoahGC -jar application.jar
```


aws

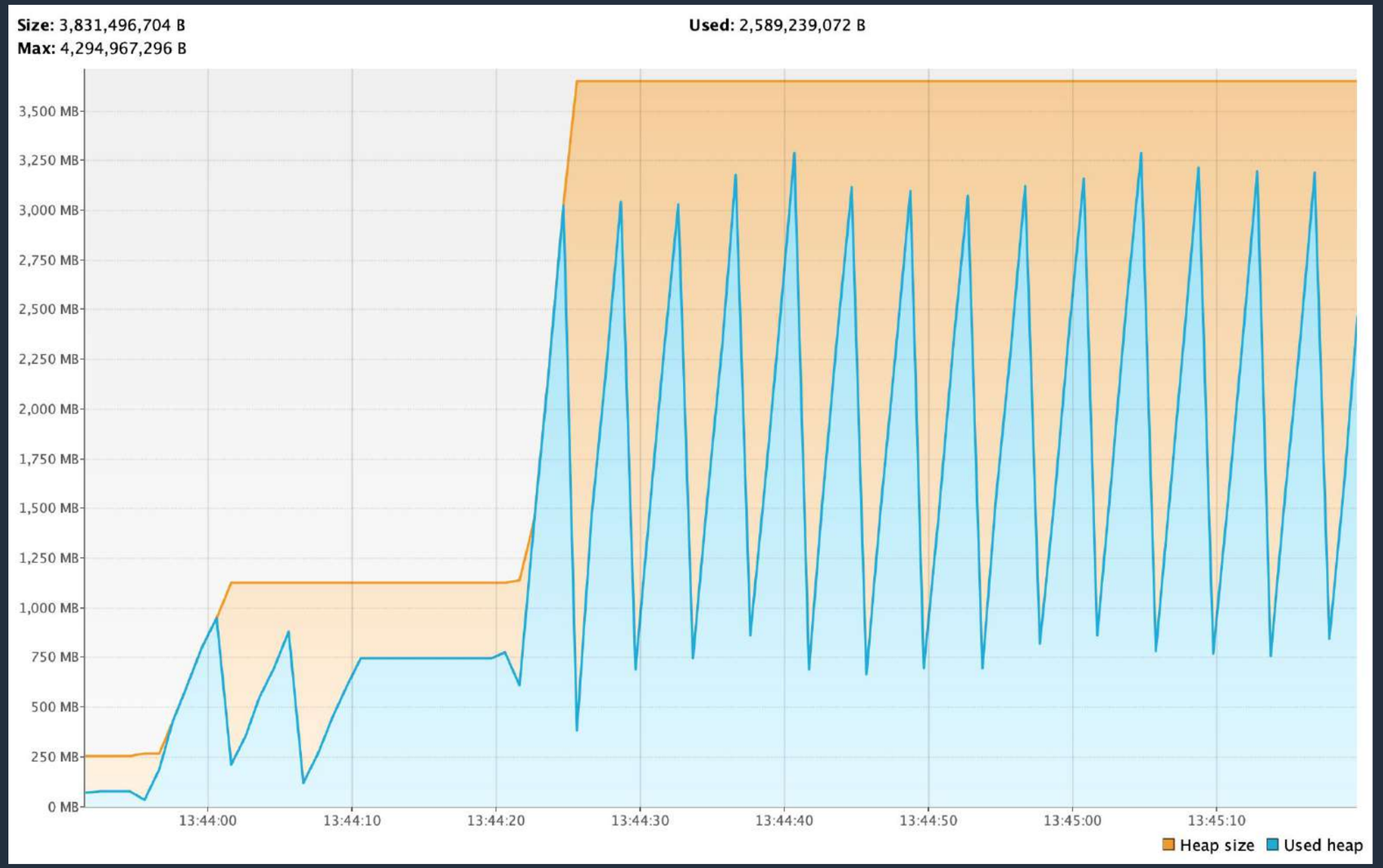
Client-side latencies



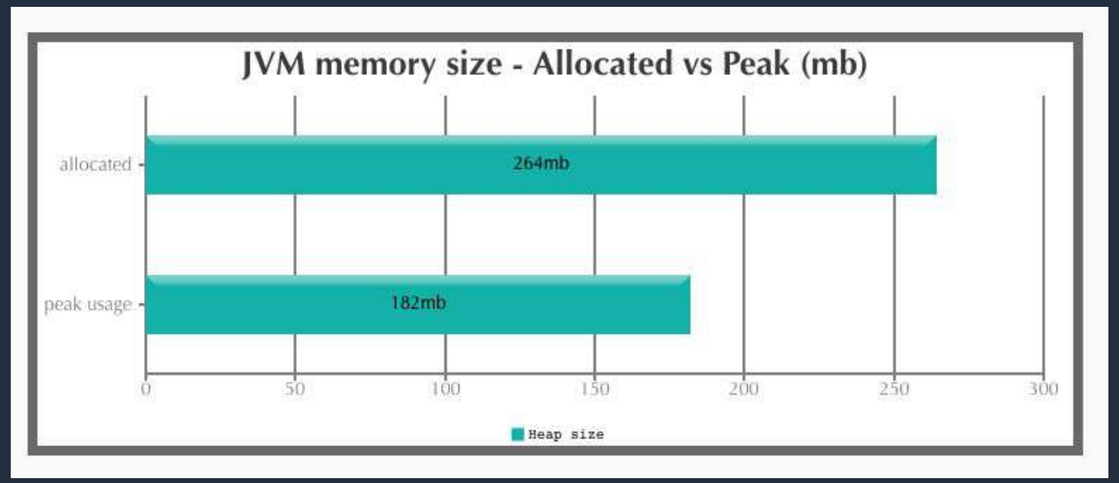
Heap statistics: OpenJDK G1



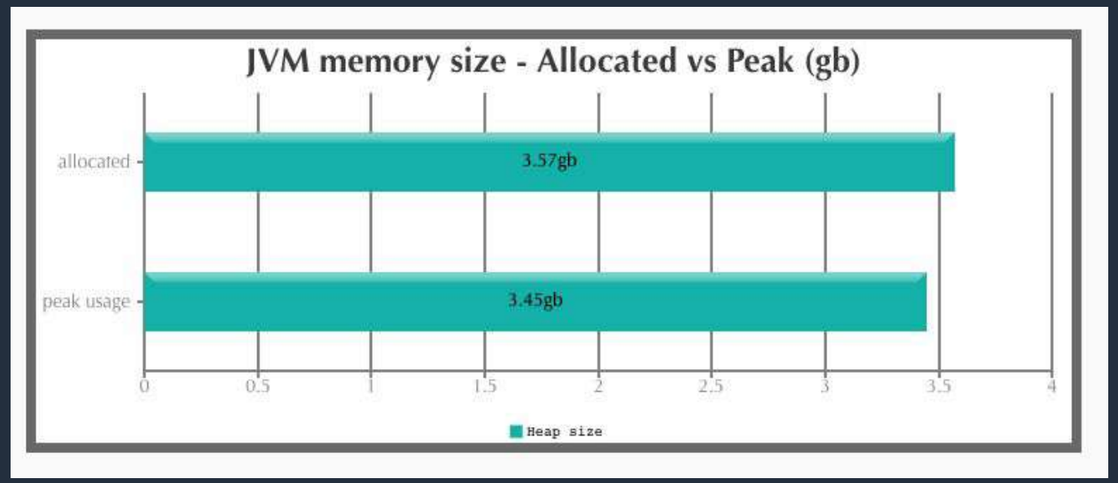
Heap statistics: Corretto Shenandoah



JVM Memory Size



OpenJDK G1



Corretto Shenandoah

aws

Key Performance Indicators: OpenJDK G1

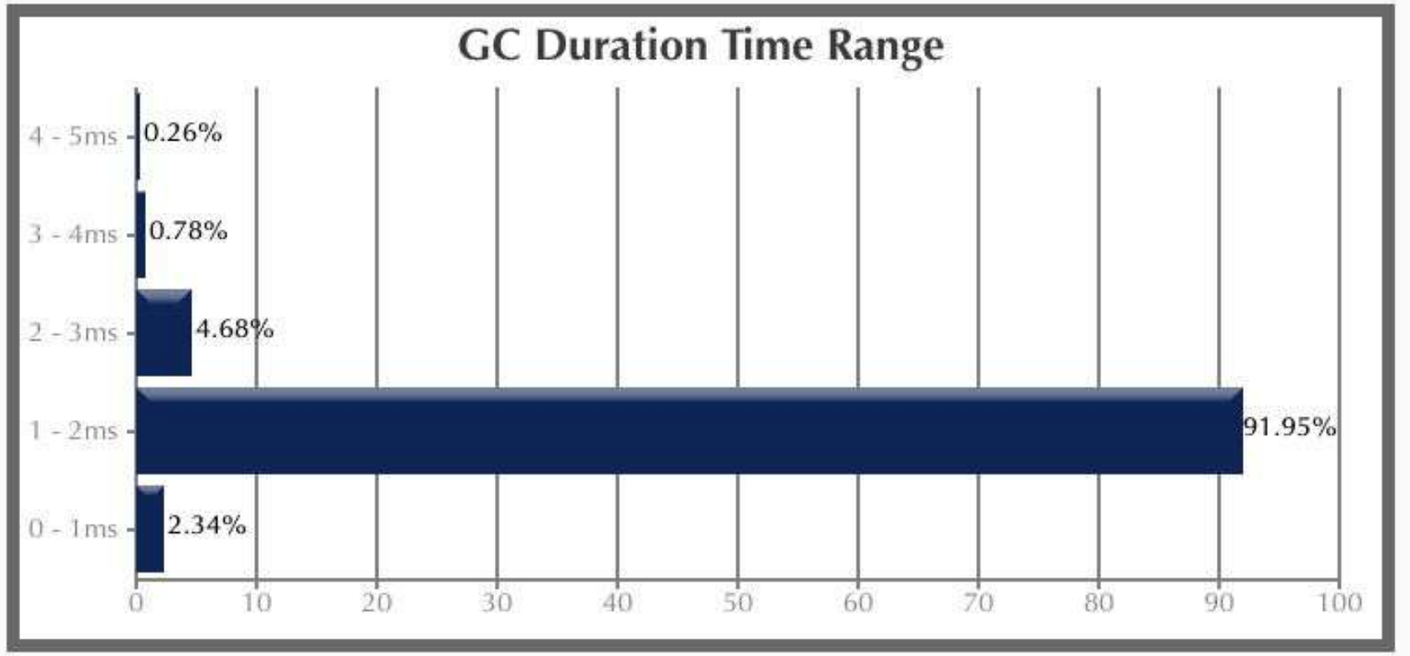
1 Throughput : 99.402%

2 Latency:

Avg Pause GC Time	1.42 ms
Max Pause GC Time	4.21 ms

GC **Pause** Duration Time Range:

Duration (ms) 1 ms Change	No. of GCs	Percentage
0 - 1	9	2.34%
1 - 2	354	91.95%
2 - 3	18	4.68%
3 - 4	3	0.78%
4 - 5	1	0.26%



Key Performance Indicators: Corretto Shenandoah

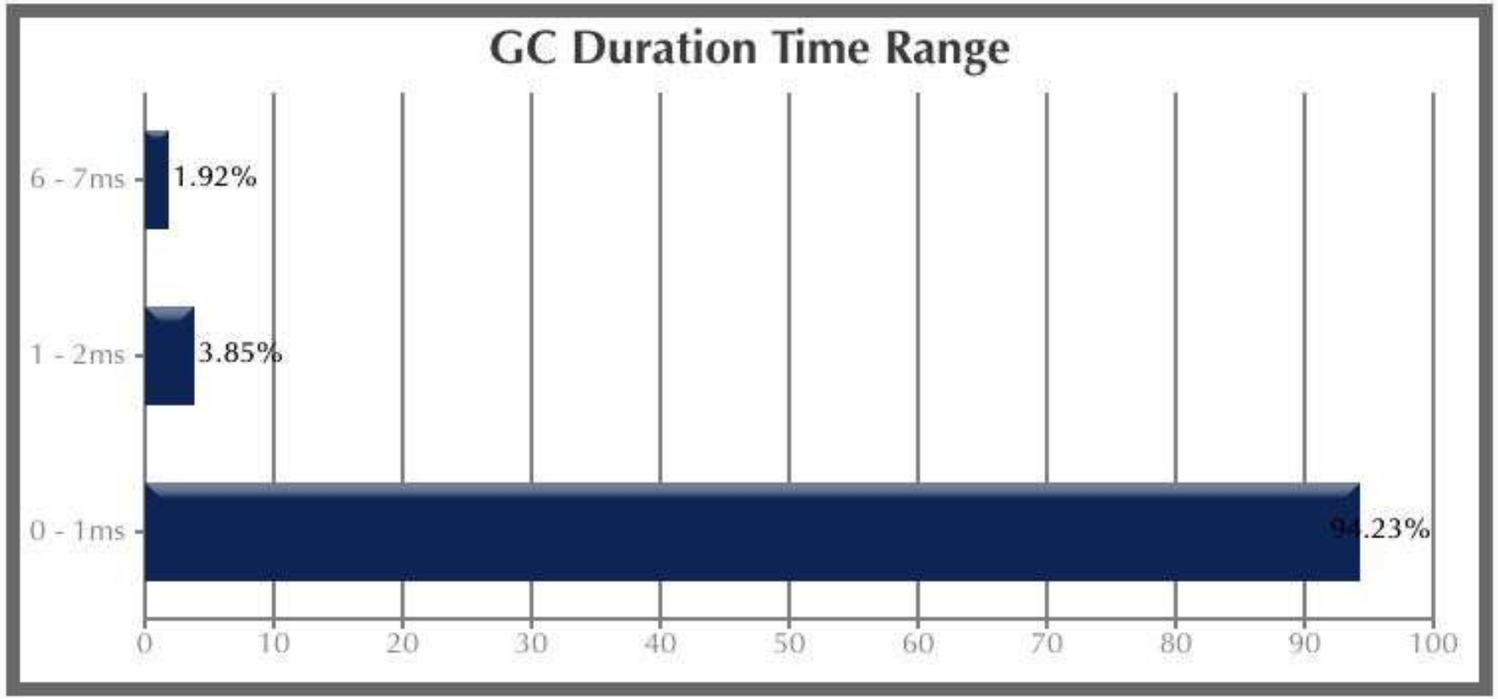
1 Throughput : 99.971%

2 Latency:

Avg Pause GC Time	0.502 ms
Max Pause GC Time	6.02 ms

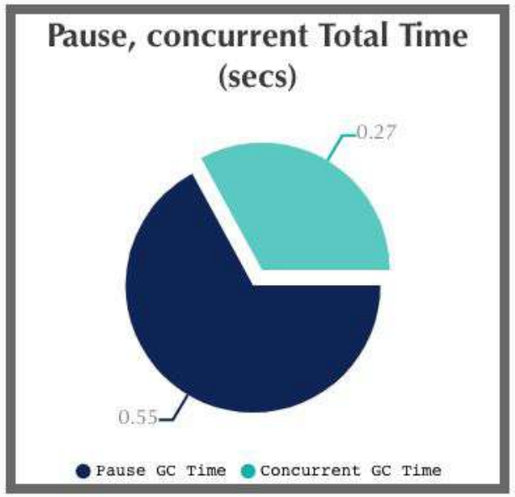
GC Pause Duration Time Range:

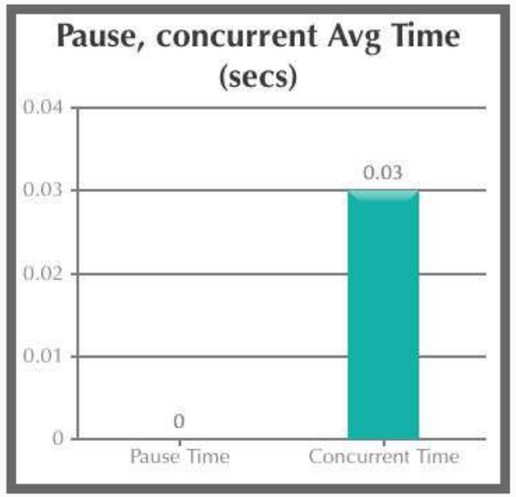
Duration (ms) 1 ms Change	No. of GCs phases	Percentage
0 - 1	49	94.23%
1 - 2	2	3.85%
6 - 7	1	1.92%



GC Pause / Concurrent times

G1 GC Time





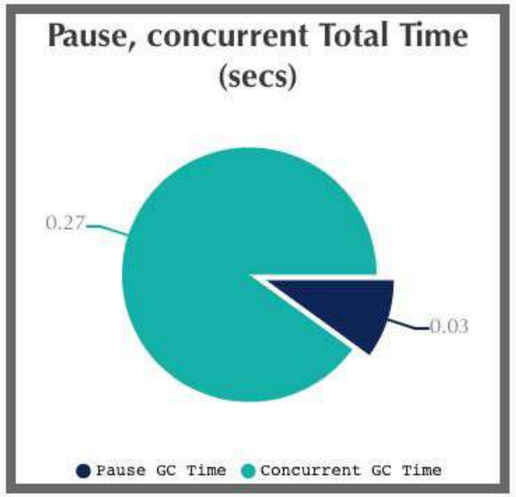
Pause Time

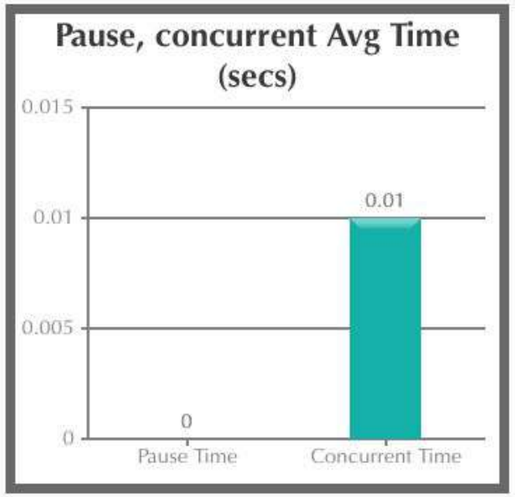
Total Time	545 ms
Avg Time	1.42 ms
Std Dev Time	0.412 ms
Min Time	0.0590 ms
Max Time	4.21 ms

Concurrent Time

Total Time	271 ms
Avg Time	33.8 ms
Std Dev Time	20.6 ms
Min Time	4.99 ms
Max Time	76.7 ms

Shenandoah GC Time





Pause Time

Total Time	26.1 ms
Avg Time	0.502 ms
Std Dev Time	0.812 ms
Min Time	0.0260 ms
Max Time	6.02 ms

Concurrent Time

Total Time	266 ms
Avg Time	14.0 ms
Std Dev Time	6.83 ms
Min Time	8.59 ms
Max Time	31.9 ms

Micronaut Community Resources

- @micronautfw
- gitter.im/micronautfw
- micronaut.io/launch
- docs.micronaut.io
- micronaut.io/learn
- guides.micronaut.io
- micronaut.io/blog
- micronaut.io/faq
- micronaut.io/foundation
- github.com/micronaut-projects/micronaut-core
- objectcomputing.com/products/micronaut/solutions
- objectcomputing.com/products/micronaut
- objectcomputing.com/resources/events

aws

Amazon Corretto Resources

https://github.com/corretto
https://aws.amazon.com/getting-started/
https://aws.amazon.com/corretto/
https://docs.aws.amazon.com/corretto/
#corretto

Questions?