

KARSTEN SILZ
4 OCTOBER 2023



WHEN IS NATIVE JAVA WITH GRAALVM WORTHWHILE FOR ME?

FOR **MOST** OF YOU:
NOT NOW

SUMMARY

JAVA MORE EXPENSIVE:
TUNED FOR LONG-LIVED
BIG APPLICATIONS

**JAVA'S NEW
COMPETITION:
JAVASCRIPT & PYTHON**

JAVA CHEAPER:

GDS, GRAC¹²³,

GRAALVM⁴⁵⁶⁷⁸

FASTER START
LESS MEMORY
SMALLER FILES
BETTER SECURITY

START SB 2 – QUARKUS

TIME (MS): 589 – 10.4

RAM (MB): 107 – 5.6

MONOLITH



MICROSERVICES



SERVERLESS



WHO MADE ME
THE EXPERT?

**EDITOR OF ARTICLE SERIES & AUTHOR
OF MULTIPLE NEWS ITEMS ON**

DATA AS EVIDENCE

JAVA EXPERTS HELPED

**NEUTRAL: NOT IN DEVELOPER
RELATIONS, NOT SELLING ANYTHING**



**BETTER
PROJECTS
FASTER**

SLIDES &
MORE



HTTPS://BPDF.LI/LAB

AGENDA

PROBLEM?

CDS & CRAC

GRAALVM

WORTH IT WHEN?

PROBLEM?

**"THE LONG-TERM PAIN POINTS OF JAVA'S
SLOW STARTUP TIME, SLOW TIME TO PEAK
PERFORMANCE, AND LARGE FOOTPRINT"**

JAVA LANGUAGE ARCHITECT

MARK REINHOLD, ORACLE, APRIL 2020

SLOW STARTUP,
SLOW TIME TO PEAK
PERFORMANCE,
LARGE FOOTPRINT

FOR 10+ YEARS

WHY?

JAVA UNDER THE HOOD

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

BUILD TIME

JVM

JIT COMPILER

BYTE-
CODE

CLASS
LIST

INIT JDK &
FRAMEWORK

INIT
APP

...

MACHINE
CODE

RUNTIME

SLOW STARTUP,
SLOW TIME TO PEAK
PERFORMANCE,
LARGE FOOTPRINT

SLOW STARTUP,
SLOW TIME TO PEAK
PERFORMANCE,
LARGE FOOTPRINT

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

BUILD TIME

RUNS EACH TIME,
SAME RESULT:
MANY JAVA OBJECTS

JVM

JIT COMPILER

BYTE-
CODE

CLASS
LIST

INIT JDK &
FRAMEWORK

INIT
APP

...

MACHINE
CODE

RUNTIME

SLOW STARTUP,
SLOW TIME TO PEAK
PERFORMANCE,
LARGE FOOTPRINT

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

BUILD TIME

PROFILING,
FAST COMPILATION (C1),
PROFILING,
OPTIMIZED COMPILATION (C2)

JVM

JIT COMPILER

BYTE-
CODE

CLASS
LIST

INIT JDK &
FRAMEWORK

INIT
APP

...

MACHINE
CODE

RUNTIME

SLOW STARTUP,
SLOW TIME TO PEAK
PERFORMANCE,
LARGE FOOTPRINT

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

BUILD TIME

PROFILER & COMPILER
BUNDLED WITH OUR APP

JVM

JIT COMPILER

BYTE-
CODE

CLASS
LIST

INIT JDK &
FRAMEWORK

INIT
APP

...

MACHINE
CODE

RUNTIME

SLOW STARTUP,
SLOW TIME TO PEAK
PERFORMANCE,
LARGE FOOTPRINT

WORKS FOR
LONG-LIVED APPS
WITH **MUCH** MEMORY

= **JAVA MONOLITHS**

TODAY



JAVA MONOLITHS

PAST

APP SERVER

**JAVA
MONOLITH**

**JAVA
MONOLITH**

NOW

CONTAINER MANAGER

JAVA
MIC.-SERV.

JAVA
MIC.-SERV.

C#
MIC.-SERV.

GO
MIC.-SERV.

JS
MIC.-SERV.

PYTHON
MIC.-SERV.

PAST

APP SERVER

JAVA
MONOLITH

JAVA
MONOLITH

NOW

CONTAINER MANAGER

JAVA
MIC.-SERV.

JAVA
MIC.-SERV.

C#
MIC.-SERV.

GO
MIC.-SERV.

JS
MIC.-SERV.

PYTHON
MIC.-SERV.

**JAVA USES MORE
MEMORY & CPU THAN
JAVASCRIPT & PYTHON**

**JAVA MORE EXPENSIVE
IN CONTAINERS/VMS**

**JAVA'S NEW
COMPETITION:
JAVASCRIPT & PYTHON**

POPULARITY?

I MEASURE

POPULARITY:

FOLLOW THE MONEY

MENTIONS IN JOB ADS
ON INDEED
(59 COUNTRIES)

PYTHON: 277.1K

PYTHON: 163.3K

JAVASCRIPT: 238.5K

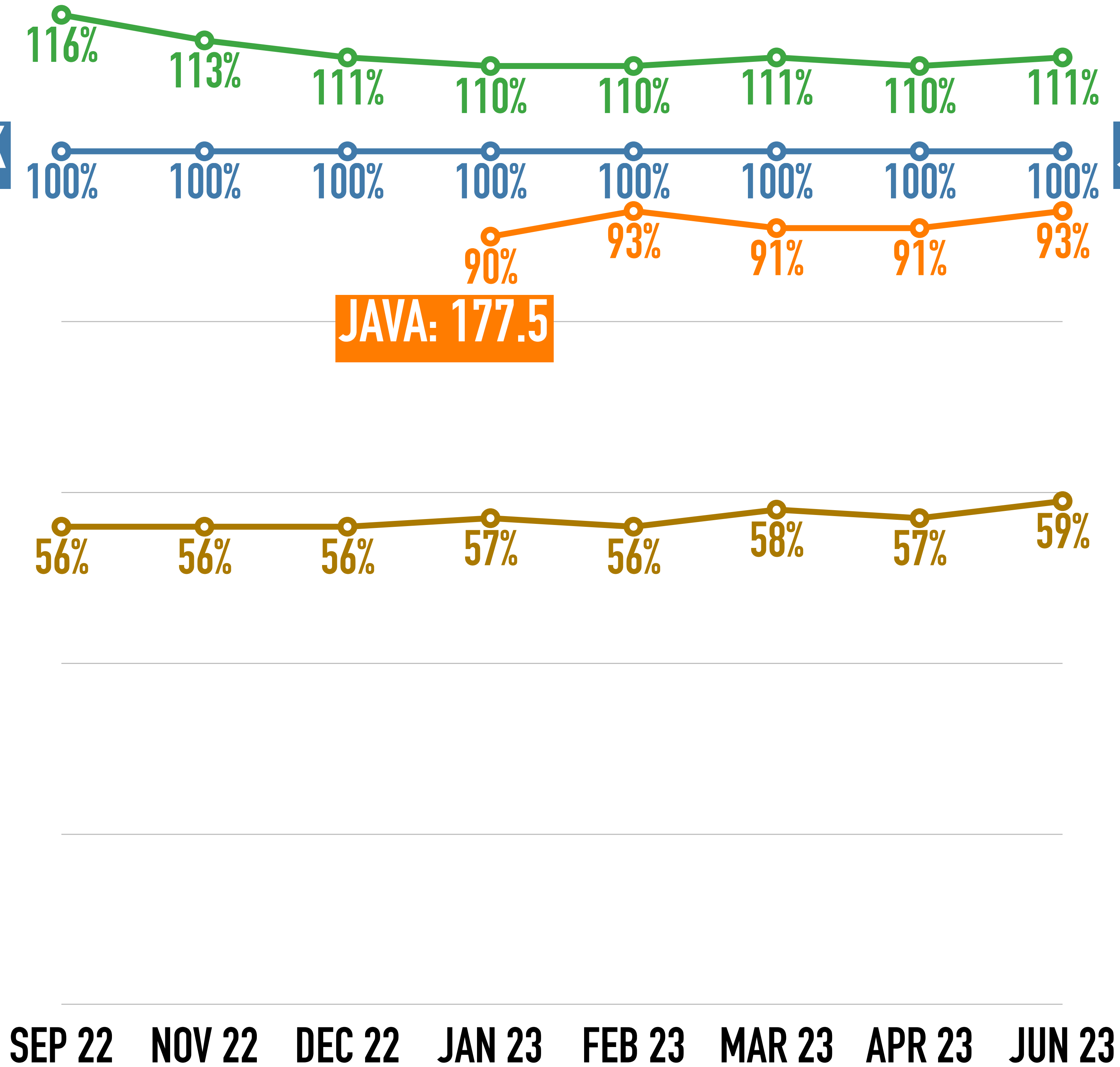
JAVASCRIPT: 147.4K

JAVA: 137.5K

JAVA: 177.5

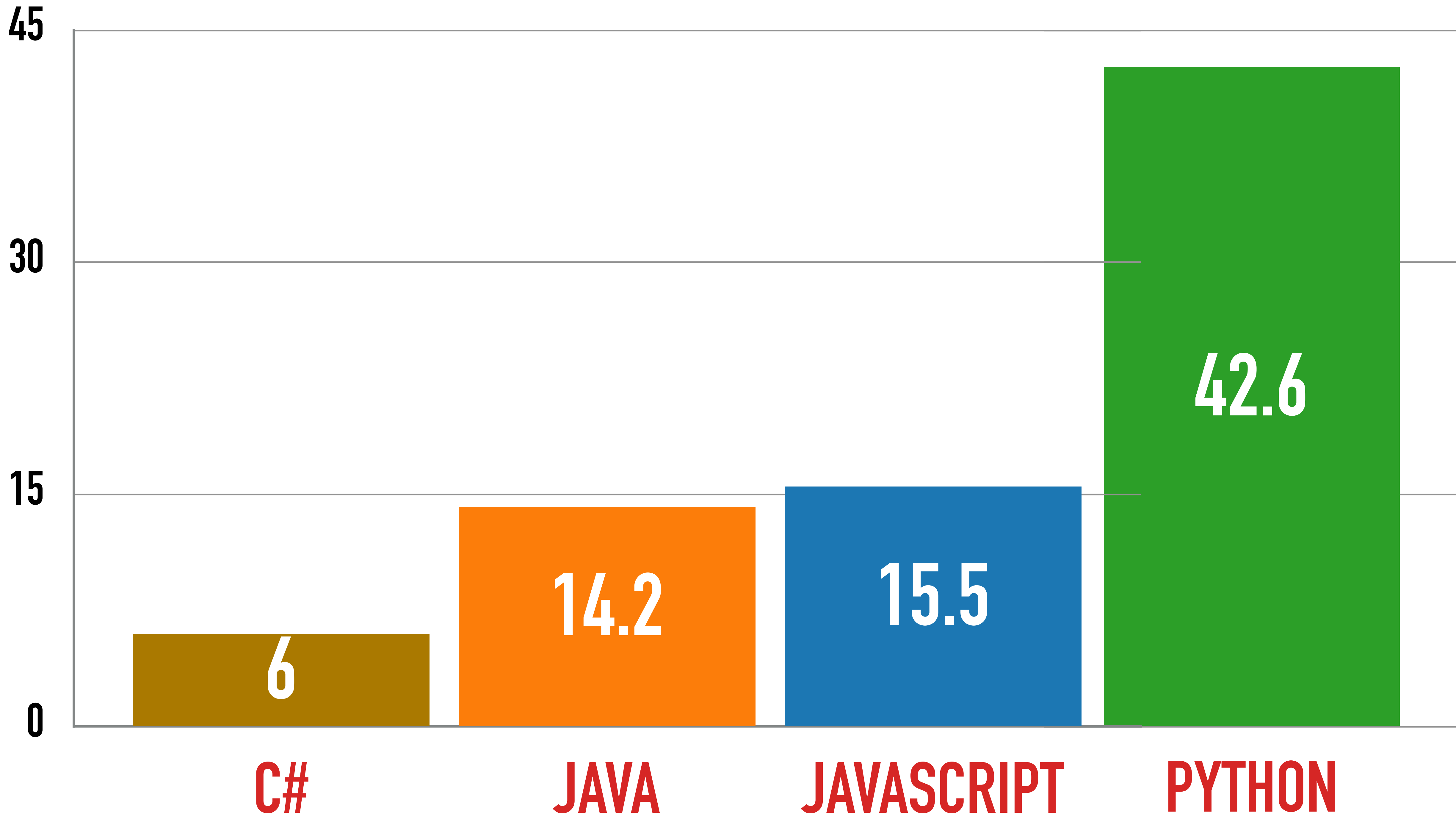
C#: 133K

C#: 86.7K



**MORE JOB ADS FOR
JAVASCRIPT & PYTHON!**

**COURSES BOUGHT
ON UDEMY
(IN MILLION)**



**BEGINNERS &
DEVELOPERS PICK
PYTHON OVER JAVA 3:1...**

...AND **3.5:1** IN
LAST 15 MONTHS!

JAVASCRIPT & PYTHON

MORE POPULAR

THAN JAVA

**SAME FOR GOOGLE
SEARCHES & STACK
OVERFLOW**

MY **FREE** NEWSLETTER:
POPULARITY OF 7 JAVA
TECHNOLOGIES

**BUT JAVA IS FASTER
FOR LONG-LIVED
APPLICATIONS!**

YES

BUT THAT OFTEN
DOES **NOT** MATTER!

DOESN'T APPLY TO
SHORT-LIVED APPS

NO JAVA DEVELOPERS:
SLOW JAVASCRIPT APP
BETTER THAN NO APP

IF APP SPENDS
90% OF TIME **WAITING**
FOR DB & APIS...

...THEN **JAVA** CAN BE
ONLY **10%** FASTER

COMPETITIVE ADVANTAGES

PYTHON:

LANGUAGE OF AI

JAVASCRIPT:

FRONT-END & BACK-END

SECTION SUMMARY

JAVA EXPENSIVE:
TUNED FOR LONG-LIVED
BIG APPLICATIONS

**JAVA'S NEW
COMPETITION:
JAVASCRIPT & PYTHON**

AGENDA

PROBLEM?

CDS & CRAC

GRAALVM

WORTH IT WHEN?

CDS & CRAC



NOT DRUGS!

**MAKE JIT JAVA
CHEAPER...**

...BY SAVING
CPU TIME

= FASTER START

SAVES MONEY

HOW?

**FEWER STANDBY
SERVERS/NODES**

POSSIBLY CHEAPER
IN SERVERLESS

LESS MEMORY
SAVES MORE MONEY
THAN FASTER START

WHY?

MORE CONTAINERS/VMS

PER MACHINE

(UNLESS CPU-LIMITED)

= LESS MACHINES

**BACK TO
CDS & CRAC**

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

BUILD TIME

RUNS EACH TIME,
SAME RESULT:
MANY JAVA OBJECTS

JVM

JIT COMPILER

BYTE-
CODE

CLASS
LIST

INIT JDK &
FRAMEWORK

INIT
APP

...

MACHINE
CODE

RUNTIME

CACHE JAVA OBJECTS

STORE THEM
ON FIRST RUN...

...AND **LOAD** THEM
ON **NEXT** RUNS

**SAVES CPU TIME
BUT NOT MEMORY**

CDS:

CCLASS **D**ATA **S**HARING

CACHES CLASS LIST

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

BUILD TIME

JVM

JIT COMPILER

BYTE-
CODE

CLASS
LIST

INIT JDK &
FRAMEWORK

INIT
APP

...

MACHINE
CODE

RUNTIME

SAVES [~]10% OF
START TIME

ENABLED WITH JAVA 17

JRE PARAMETERS

CRAC:

COORDINATED RESTORE

AT CHECKPOINT

CACHES

FULL APPLICATION

SNAPSHOT

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

BUILD TIME

JVM

JIT COMPILER

BYTE-
CODE

CLASS
LIST

INIT JDK &
FRAMEWORK

INIT
APP

...

MACHINE
CODE

RUNTIME

INCLUDES
MACHINE CODE

APP DECIDES WHEN
TO TAKE SNAPSHOT

CRAC START TIME:

QUARKUS: 1S \Rightarrow 46 MS

SPRING BOOT: 3.9 S \Rightarrow 38 MS

JAVA CHEAPER:

GDS, GRAC¹²³,

GRAALVM⁴⁵⁶⁷⁸

1 AZUL OPENJDK

2 FRAMEWORK

3 LINUX ONLY

CATCH #1:

AZUL OPENJDK

ONLY **AZUL** OPENJDK
SUPPORTS CRAC

FREE FOR PRODUCTION

CATCH #2: FRAMEWORK

SPRING BOOT 3.2

(NOV 23) / QUARKUS /

MICRONAUT

CATCH #3: LINUX ONLY

CRUI:

**CHECKPOINT RESTORE
IN USERSPACE**

SECTION SUMMARY

JAVA CHEAPER:

CDS, GRAC¹²³,

GRAALVM⁴⁵⁶⁷⁸

AGENDA

PROBLEM?

CDS & CRAC

GRAALVM

WORTH IT WHEN?

GRAALVM

**MAKES NATIVE JAVA
CHEAPER...**

...BY SAVING

CPU TIME & MEMORY

THREE GRAALVM PROJECTS

GRAALVM NATIVE IMAGE

COMPILER

PART OF OPENJDK

SINCE 2023

RELEASES WITH JAVA
SINCE JAVA 21

NATIVE JAVA?

"NATIVE JAVA" =

"STATIC JAVA" =

"AOT JAVA"

"GRAALVM"

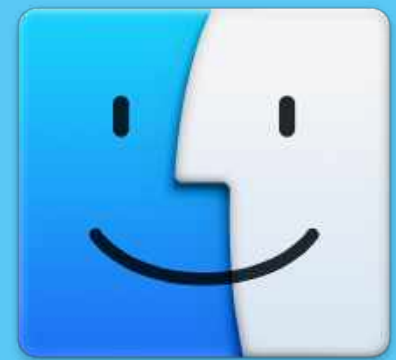
FROM 1 BYTECODE JAR...

```
java -jar my-app-1.2.jar
```

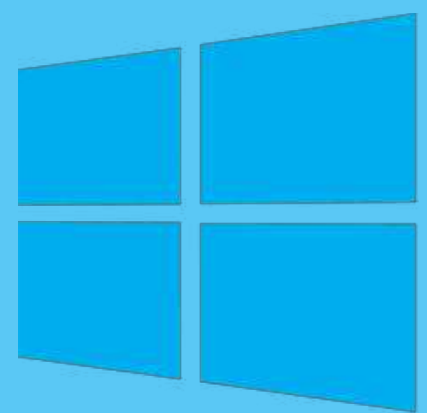
...TO 3 EXECUTABLES



`./my-app-1.2-runner`



`./my-app-1.2-runner`



`my-app-1.2-runner.exe`

DEMO MICROSERVICE
FOR THIS TALK

80 PICTURES TO PDF 5X:

**SPRING BOOT /
QUARKUS / GO**

SAVINGS WITH
NATIVE IMAGE

START SB 2 – QUARKUS

TIME (MS): 589 – 10.4

RAM (MB): 107 – 5.6

START SB 2 – QUARKUS – GO

TIME (MS): 589 – 10.4 – 2.6

RAM (MB): 107 – 5.6 – 2.8

HOW?

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

BUILD TIME

JVM

JIT COMPILER

BYTE-
CODE

CLASS
LIST

INIT JDK &
FRAMEWORK

INIT
APP

...

MACHINE
CODE

RUNTIME

**MOVE WORK
TO BUILD TIME**

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

BUILD TIME

JVM

JIT COMPILER

BYTE-
CODE

CLASS
LIST

INIT JDK &
FRAMEWORK

INIT
APP

...

MACHINE
CODE

RUNTIME

JAVA COMPILER

**SOURCE
CODE**

**BYTE-
CODE**

JIT COMPILER

**MACHINE
CODE**

BUILD TIME

JVM

**BYTE-
CODE**

**CLASS
LIST**

**INIT JDK &
FRAMEWORK**

**INIT
APP**

...

RUNTIME

JAVA COMPILER

**SOURCE
CODE**

**BYTE-
CODE**

BUILD TIME

**AOT: AHEAD-OF-TIME
(LIKE C++-COMPILER)**

AOT COMPILER

**MACHINE
CODE**

JVM

**BYTE-
CODE**

**CLASS
LIST**

**INIT JDK &
FRAMEWORK**

**INIT
APP**

...

RUNTIME

JAVA COMPILER

**SOURCE
CODE**

**BYTE-
CODE**

AOT COMPILER

**MACHINE
CODE**

BUILD TIME

JVM

**MACHINE
CODE**

**CLASS
LIST**

**INIT JDK &
FRAMEWORK**

**INIT
APP**

...

RUNTIME

JAVA COMPILER

AOT COMPILER

**SOURCE
CODE**

**BYTE-
CODE**

**CLASS
LIST**

**MACHINE
CODE**

BUILD TIME

JVM

**MACHINE
CODE**

**INIT JDK &
FRAMEWORK**

**INIT
APP**

RUNTIME

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

CLASS
LIST

AOT COMPILER

MUCH INIT JDK &
FRAMEWORK

MUCH INIT
APP

MACHINE
CODE

BUILD TIME

JVM

MACHINE
CODE

LITTLE INIT JDK &
FRAMEWORK

LITTLE INIT
APP

RUNTIME

JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

CLASS
LIST

GRAALVM NATIVE IMAGE

MUCH INIT JDK &
FRAMEWORK

MUCH INIT
APP

MACHINE
CODE

BUILD TIME

SUBSTRATE VM

MACHINE
CODE

LITTLE INIT JDK &
FRAMEWORK


LITTLE INIT
APP

RUNTIME

FASTER START
LESS MEMORY
SMALLER FILES
BETTER SECURITY

CWA

CWA?

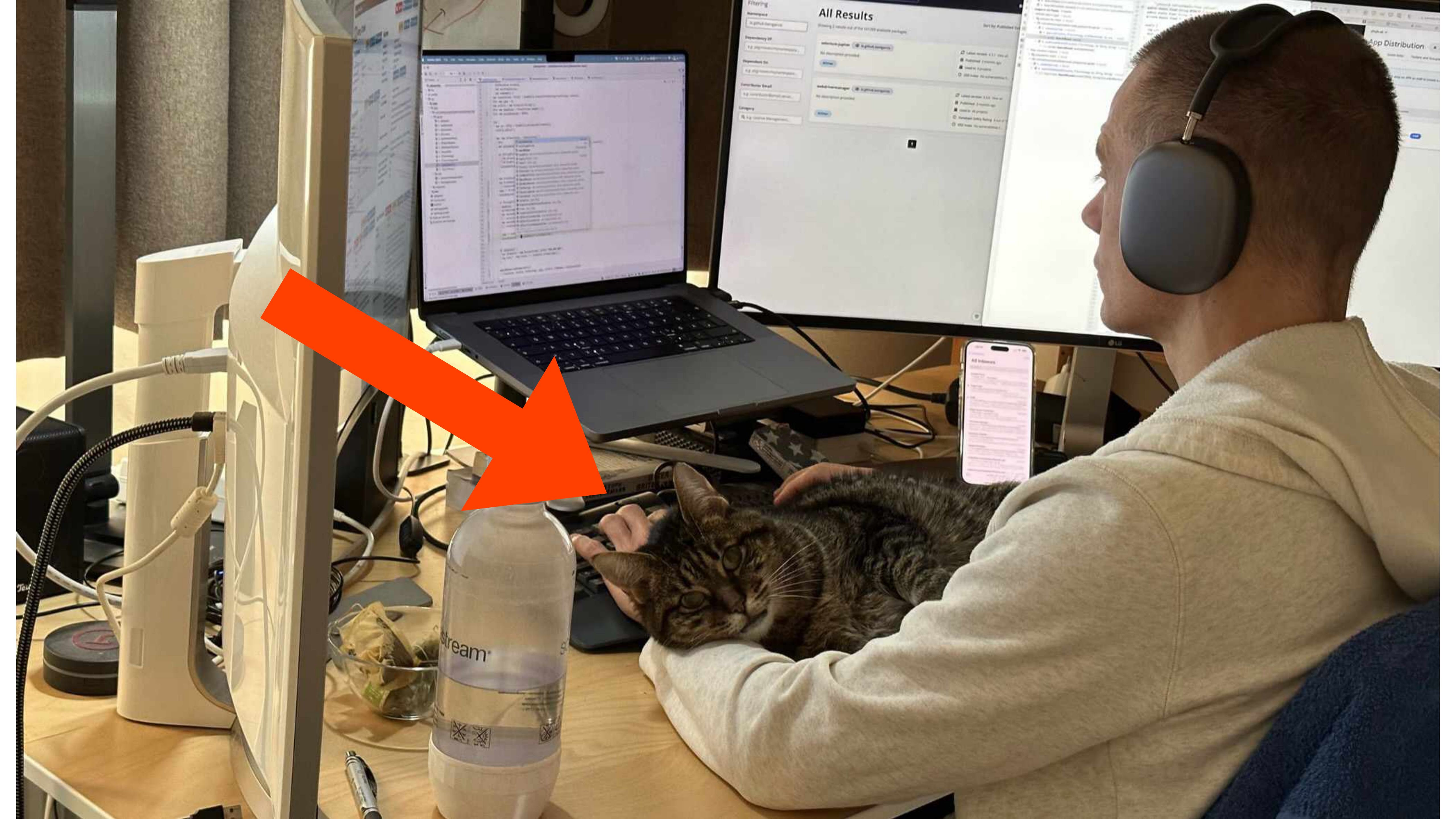
A photograph of two male wrestlers in a blue and red singlet embracing on a wrestling mat. The background is blurred, showing spectators and arena lighting. The text 'CZECH WRESTLING ASSOCIATION?' is overlaid in large white letters.

CZECH WRESTLING ASSOCIATION?

How and Why to Add Cats
to Your Fiction Writing



CAT WRITING
ASSOCIATION?





JAVA COMPILER

SOURCE
CODE

BYTE-
CODE

CLASS
LIST

GRAALVM NATIVE IMAGE

MUCH INIT JDK &
FRAMEWORK

MUCH INIT
APP

MACHINE
CODE

BUILD TIME

SUBSTRATE VM

MACHINE
CODE

LITTLE INIT JDK &
FRAMEWORK

LITTLE INIT
APP

RUNTIME

CWA: AT BUILD TIME

ALL CLASSES KNOWN

OUR APPLICATION, OUR
LIBRARIES, FRAMEWORK & JDK

ALL RESSOURCES KNOWN

ALL FILES WE WANT TO LOAD -
PROPERTY FILES, RESOURCE
BUNDLES, ETC.

NATIVE IMAGE

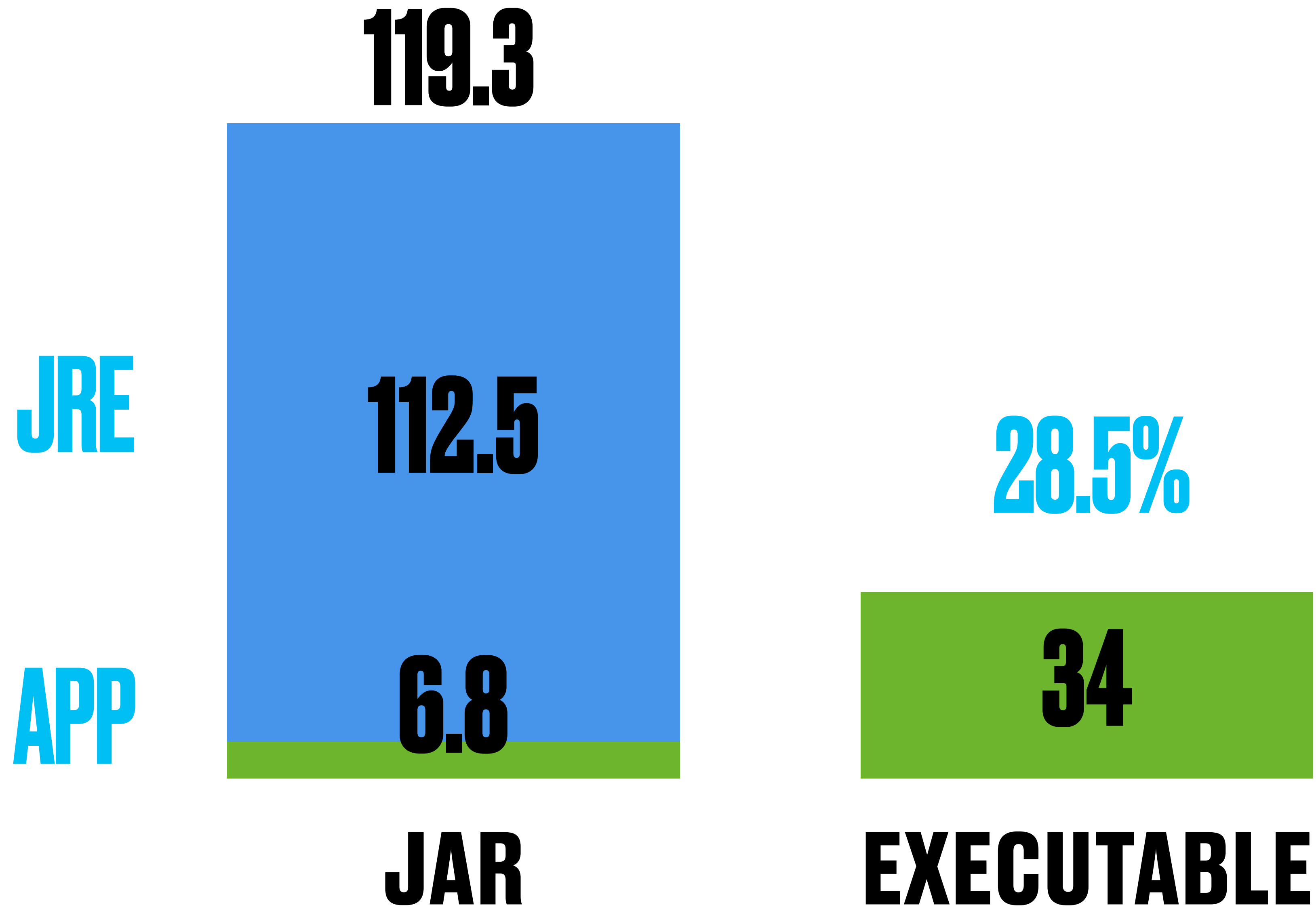
REMOVES "ALL

UNNECESSARY STUFF"

FASTER START
LESS MEMORY
SMALLER FILES
BETTER SECURITY

SMALLER FILES

QUARKUS: FILE SIZE (MB)



BETTER SECURITY

~~LOGASHIELD~~

BETTER SECURITY

SMALLER ATTACK SURFACE

REMOVING UNUSED CODE ALSO
REMOVES POTENTIAL SECURITY
HOLDS IN FRAMEWORKS &
LIBRARIES

ARBITRARY CODE DOESN'T RUN

ATTACKS LIKE LOG4SHELL GET JAVA TO
LOAD & RUN ATTACK CODE

CWA STOPS THIS: CODE MUST BE
KNOWN AT BUILD TIME

FASTER STARTUP

LESS MEMORY

SMALLER FILES

BETTER SECURITY

JAVA CHEAPER:

CDS, CRAC¹²³,

GRAALVM⁴⁵⁶⁷⁸

4 NOT ALL JAVA WORKS

5 FRAMEWORK

6 PERFORMANCE

7 MORE DEVELOPER TIME

8 HARDER HIRING

CATCH #4:

NOT ALL JAVA WORKS

POSSIBLE OR NOT?

ALWAYS IMPOSSIBLE

CREATE NEW
CLASSES & METHODS
AT RUNTIME

RUN ARBITRARY
BYTECODE

PARTIALLY/ONGOING WORK

AWT (NOT ON MACOS)

JMX

JAVA FLIGHT RECORDER

TEST FRAMEWORKS

ALWAYS POSSIBLE

REFLECTION & CLASS
LOADING - REQUIRES
CONFIGURATION

SUBSTRATE VM: GARBAGE
COLLECTION, THREAD
MANAGEMENT

MISSING CLASS =
RUNTIME CRASH

LIBRARIES DO
IMPOSSIBLE THINGS
TOO...

...AND NEED

CONFIGURATION!

CATCH #5: FRAMEWORK

GRAALVM RECOMMENDS:

USE FRAMEWORK

**BUT SOME WANT NO
FRAMEWORK OR USE
HOMEGROWN ONE**

FRAMEWORKS WITH GRAALVM PRODUCTION SUPPORT

WHY FRAMEWORKS?

CONFIGURE NATIVE IMAGE &
LIBRARIES

SHIP WITH COMPATIBLE LIBRARIES

ADDITIONAL FEATURES (LIKE HOT
CODE RELOAD IN QUARKUS)

WHICH FRAMEWORKS?

NEW: QUARKUS & MICRONAUT
– NOT HELIDON, NO JOB DEMAND

OLD: SPRING BOOT 3 (NOV 2022)

JAKARTA EE: NO

CATCH #6: PERFORMANCE

PEAK PERFORMANCE

OFTEN WORSE THAN

JIT JAVA

PEAK PERFORMANCE

GRAALVM COMMUNITY EDITION

SERIAL GARBAGE COLLECTOR: STOP THE WORLD, SINGLE-THREADED

ONLY GLOBAL OPTIMIZATIONS

ORACLE GRAALVM FOR JAVA 17 & 21

G1 GARBAGE COLLECTOR: NOT "STOP THE WORLD", MULTI-THREADED (LINUX ONLY)

**PROFILE-GUIDED OPTIMIZATION (PGO):
NATIVE IMAGE INSTRUMENTS EXECUTABLE
FOR PROFILING, USES DATA LATER FOR
COMPILATION**

ORACLE GRAALVM FOR
JAVA 17 & 21 FREE FOR
PRODUCTION...

...**BUT HAS NEW LICENSE**

CATCH #7:

MORE DEVELOPER TIME

MORE DEVELOPER TIME

CONFIGURATION

REFLECTION & CLASS
LOADING: MANUALLY OR
WITH TRACING AGENT
(MONITORS JIT JAVA)

POSSIBLY DISABLE BUILD
TIME INIT

BUILD & DEPLOYMENT

BUILDS TAKE MUCH LONGER,
CREATE 1 EXECUTABLE PER
PLATFORM (NO CROSS-
COMPILIERUNG)

DEBUGGING ON MACOS &
WINDOWS TAKES MUCH
LONGER (LINUX DETOUR)

MONITORING & PRODUCTION

SOME OBSERVABILITY
TOOLS DON'T RUN AT ALL,
OTHER ONES GIVE LESS
DATA

TROUBLESHOOTING TAKES
LONGER

LESS TRAINING
MATERIALS, LESS
ANSWERS

DEVELOPING FOR NATIVE JAVA

MOST OF THE TIME

DEVELOPERS USE JIT JAVA - AS USUAL

CI/CD PIPELINE BUILDS NATIVE JAVA EXECUTABLES

RARELY

DEVELOPERS BUILD NATIVE JAVA EXECUTABLES THEMSELVES (E.G., BEFORE MERGE OR FOR NEW LIBRARIES)

DEBUGGING NATIVE JAVA ON MAC & WINDOWS STILL ANNOYING

**CATCH #8:
HARDER HIRING**

NATIVE JAVA:

LOW JOB DEMAND

(INDEED UK, SEP 2023)

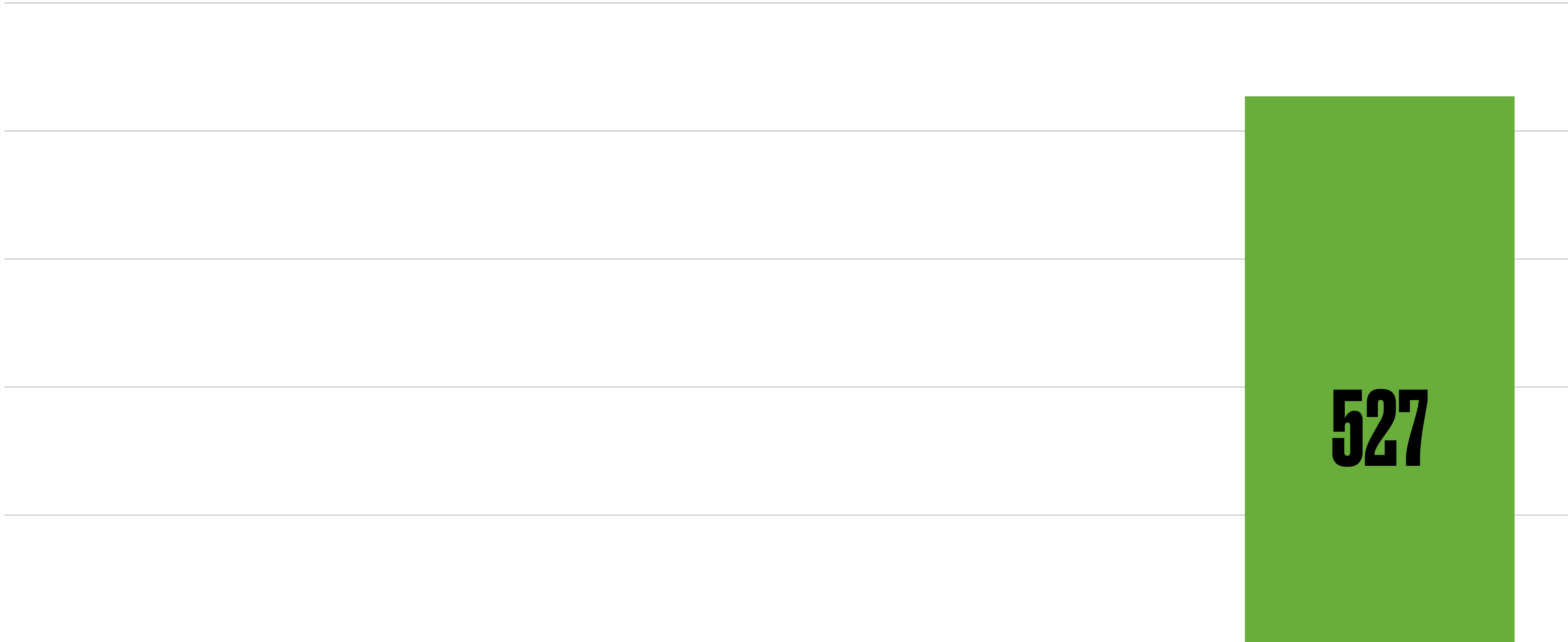
0
HELIDON

29
QUARKUS

38
MICRONAUT

25
JAKARTA EE

527
SPRING BOOT



HELIDON + QUARKUS +

MICRONAUT =

13% OF SPRING BOOT

INDIGATES LOW SUPPLY
OF GRAALVM EXPERTS

**HARD TO CONVINCE
BOSS & TEAMMATES**

SECTION SUMMARY

FASTER START
LESS MEMORY
SMALLER FILES
BETTER SECURITY

JAVA CHEAPER:

CDS, CRAC¹²,

GRAALVM³⁴⁵⁶⁷

START SB 2 – QUARKUS

TIME (MS): 589 – 10.4

RAM (MB): 107 – 7.3

GRAALVM

AGENDA

PROBLEM?

CDS & CRAC

GRAALVM

WORTH IT WHEN?

WORTH IT WHEN?

MY "FIVE STEP PLAN
FOR NATIVE JAVA"

EVERY STEP:

"GO/NO GO" DECISION

- 1. DOES THE BOSS CARE?**
- 2. DO THE NUMBERS ADD UP?**
- 3. COULD IT WORK?**
- 4. START SMALL**
- 5. GO BIG**

**DOES THE BOSS
CARE?**

OUR JOB:

CREATE BUSINESS VALUE

...**NOT** BOASTING
ABOUT MILLISECOND
START TIMES!

DOES THE BOSS CARE?

OUR JOB

CREATE BUSINESS VALUE – NEW FEATURES, BUG FIXES

"BUSINESS VALUE" NOT "SAVING MONEY"

BOSS DOESN'T CARE BECAUSE...

NEW FEATURES MORE IMPORTANT NOW

THEIR BONUS DOESN'T DEPEND ON SAVINGS IN IT DEPARTMENT

COST SAVINGS DON'T CHANGE IT BUDGET



1. DOES THE BOSS CARE?

2. DO THE NUMBERS ADD UP?

3. COULD IT WORK?

4. START SMALL

5. GO BIG

**DO THE NUMBERS
ADD UP?**

CATCH #6:

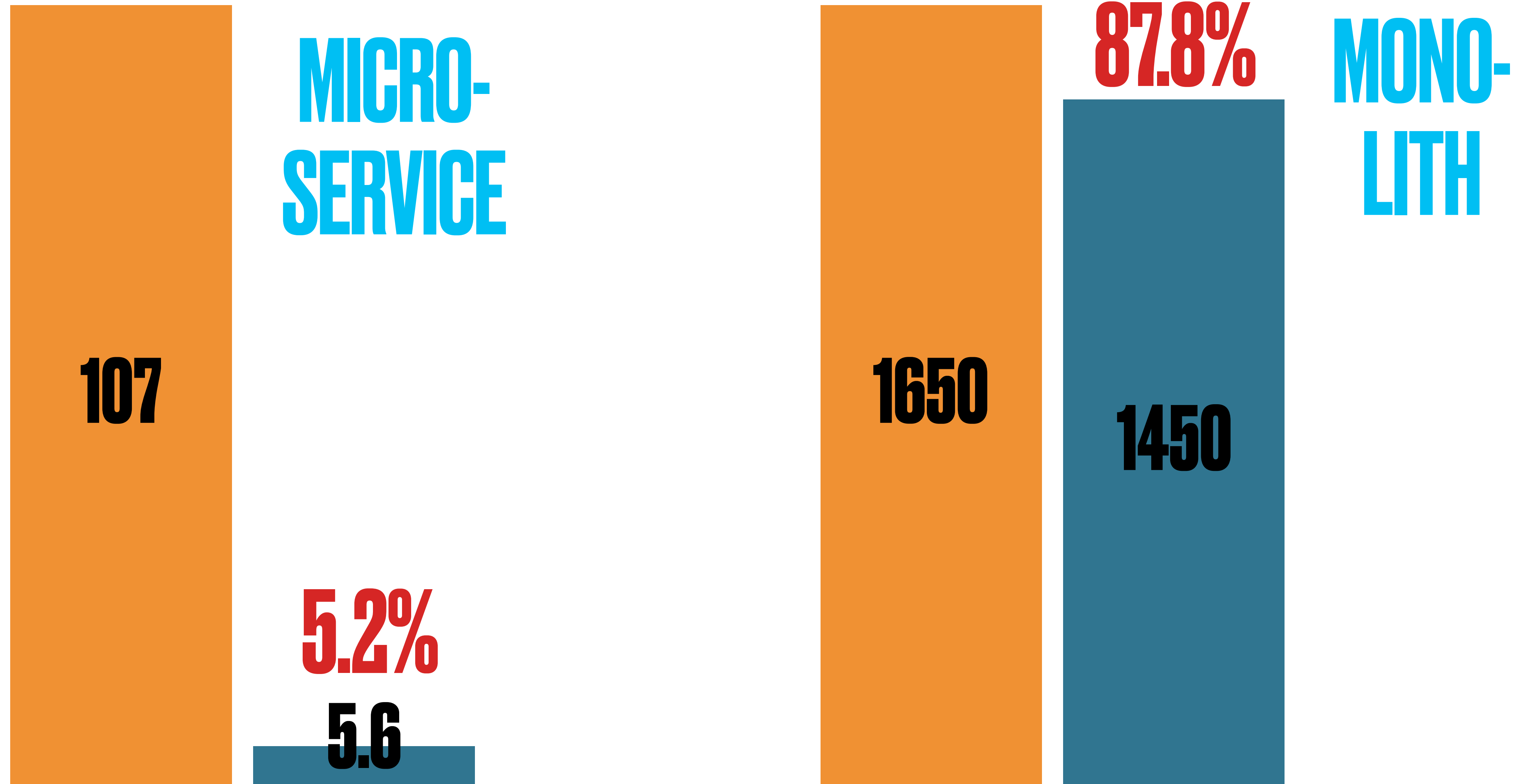
MORE DEVELOPER TIME

BAD FOR NATIVE JAVA:

DEVELOPER TIME GETS
MORE EXPENSIVE...

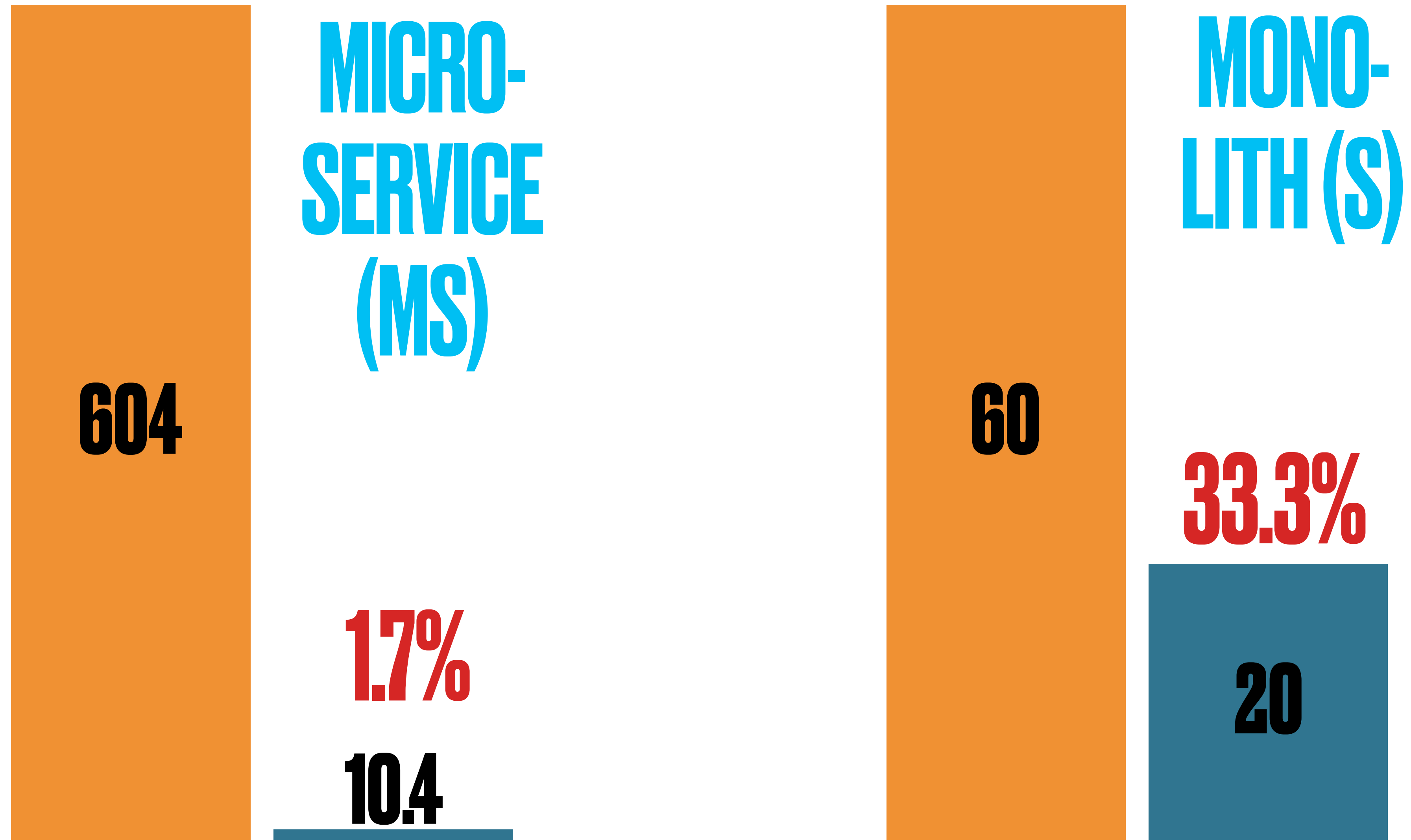
...WHILE HARDWARE
GETS CHEAPER

JIT JAVA VS. NATIVE JAVA: MEMORY



200 MB LESS IN
1.6 GB HEAP SPACE?

JIT JAVA VS. NATIVE JAVA: START TIME



40 SECONDS LESS START
TIME EVERY **2 WEEKS?**

MONOLITH =
MANY LIBRARIES =
HIGHER RISK OF
"DOESN'T WORK"

MONOLITH



MICROSERVICES



SERVERLESS



MORE CONTAINERS/VMS

PER MACHINE

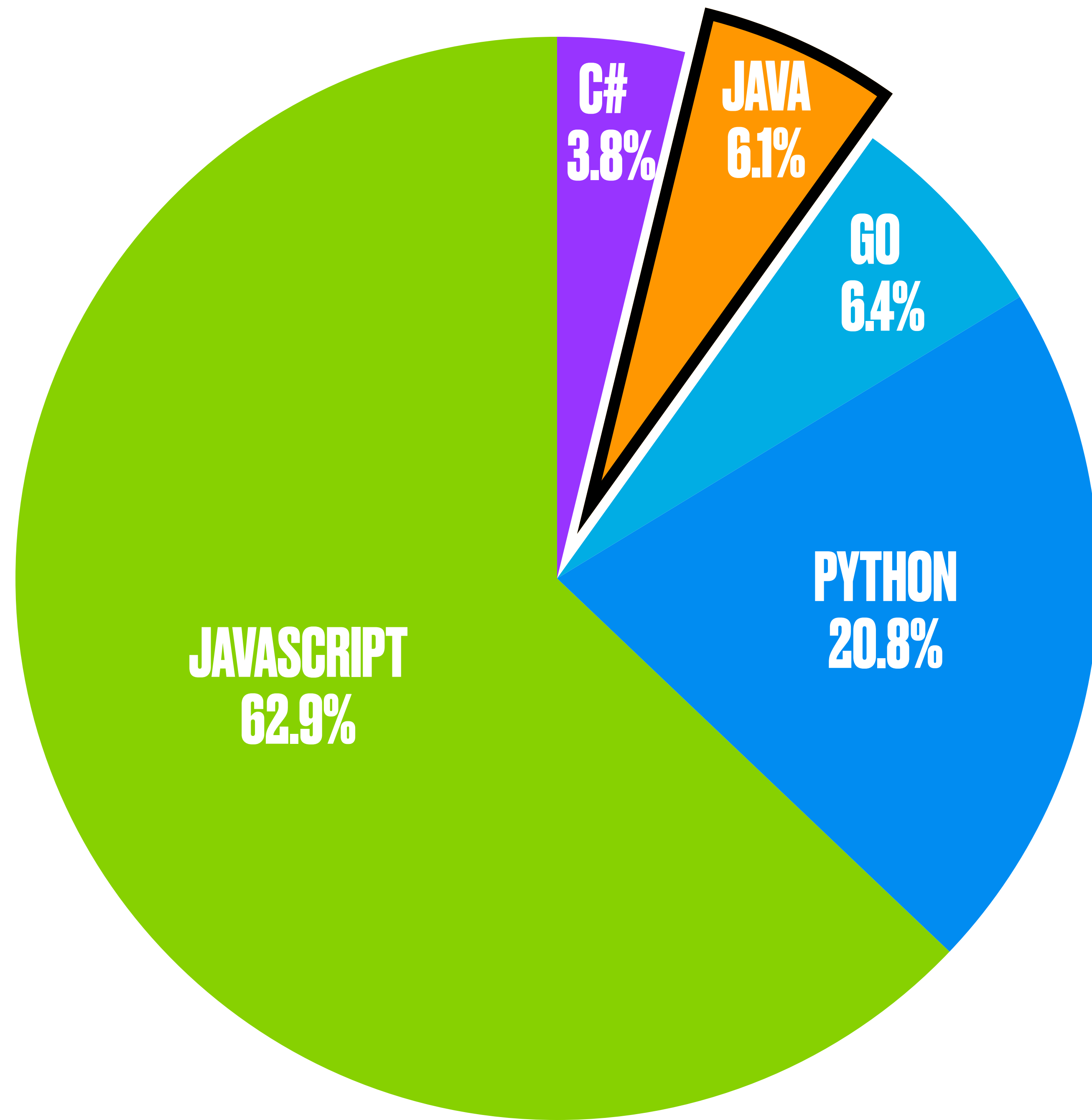
(UNLESS CPU-LIMITED)

ANNUAL SAVINGS:

10K – OR 10M?

JAVA IN SERVERLESS?

**RED HAT KEYNOTE @
DEVOXX UK, MAY 2022**



LOW SHARE BECAUSE
YOU CAN **NOT** USE JAVA
WELL IN SERVERLESS...

...OR BECAUSE **SAVINGS**
FOR CPU & MEMORY
ARE **NOT USED** YET?

I DON'T KNOW

MONOLITH



MICROSERVICES



SERVERLESS



COSTS OF NATIVE JAVA

ONE-TIME

**NEW FRAMEWORK (QUARKUS, MICRONAUT)
OR UPGRADE (SPRING BOOT 3: JAVA 17,
JAKARTA EE 9) – LIBRARIES!**

TRAIN HOW MANY DEVELOPERS?

**DEVELOPER PCS, BUILD PIPELINE, CI/CD,
OBSERVABILITY, ...**

RECURRING

**TROUBLESHOOTING TAKES
LONGER, LESS ANSWERS,
OBSERVABILITY WORSE**

**FEWER DEVELOPERS WITH
NATIVE JAVA EXPERIENCE**

THE **SMALLER** THE APP
SIZE & RUN TIME,
THE **BETTER**

ESTIMATE —

WE DON'T KNOW



1. DOES THE BOSS CARE?



2. DO THE NUMBERS ADD UP?

3. COULD IT WORK?

4. START SMALL

5. GO BIG

COULD IT WORK?

COULD IT WORK

WHAT WE KNOW

DO OUR APPLICATIONS DO THINGS
THAT DON'T WORK IN NATIVE
JAVA?

DOES THE OBSERVABILITY WORK
IN NATIVE JAVA?

WHAT WE DON'T KNOW

DO THE LIBRARIES WORK IN
NATIVE JAVA?



1. DOES THE BOSS CARE?



2. DO THE NUMBERS ADD UP?



3. COULD IT WORK?

4. START SMALL

5. GO BIG

START SMALL

START SMALL

FOR MICROSERVICES

START NEW/RE-IMPLEMENT IN
NATIVE JAVA

SMALL TEAM, LOW COSTS, LOW RISK

GOAL: DO THE NUMBERS STILL ADD
UP?

FOR MONOLITHS

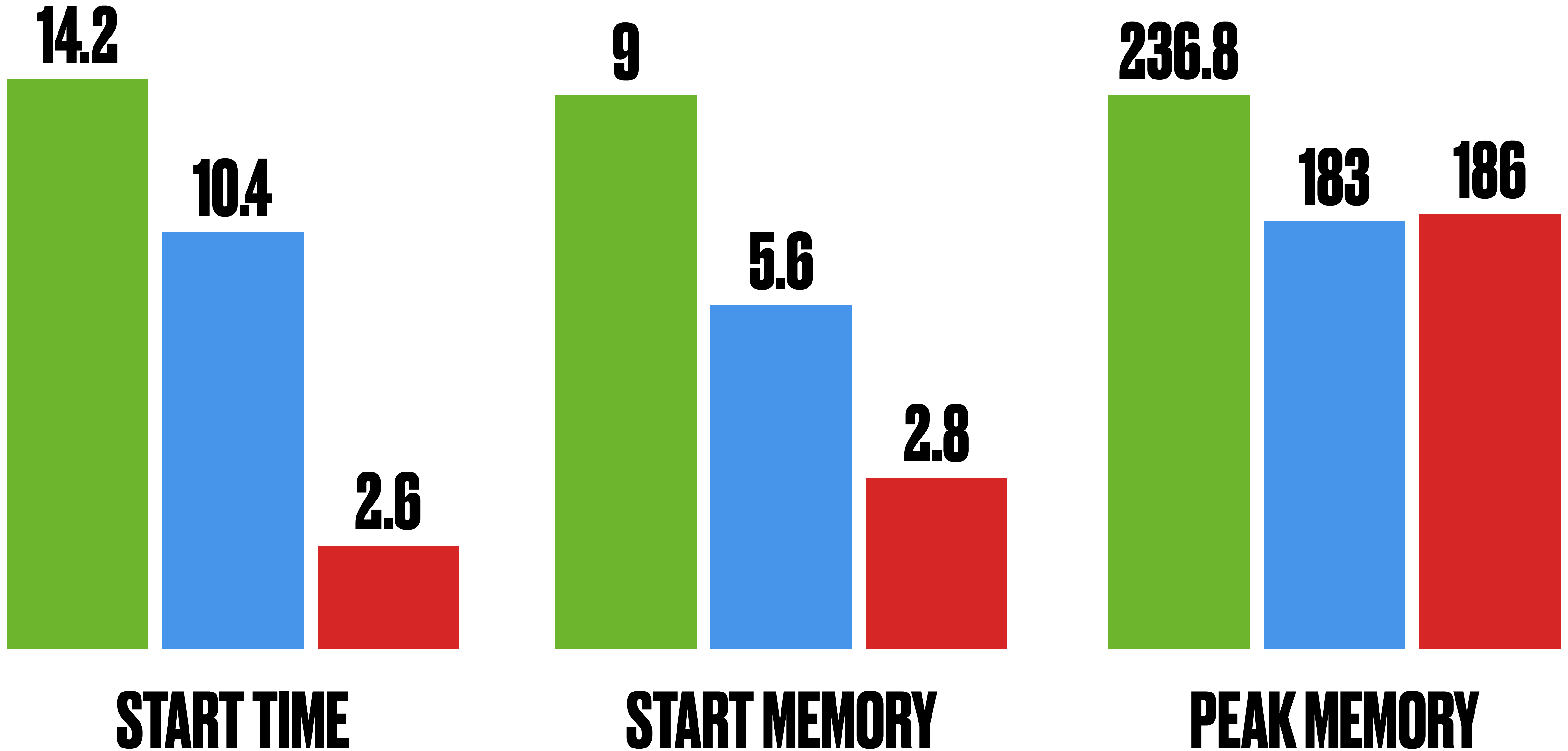
BREAK OFF A SMALL PART AS
MICROSERVICE

THEN: SEE LEFT

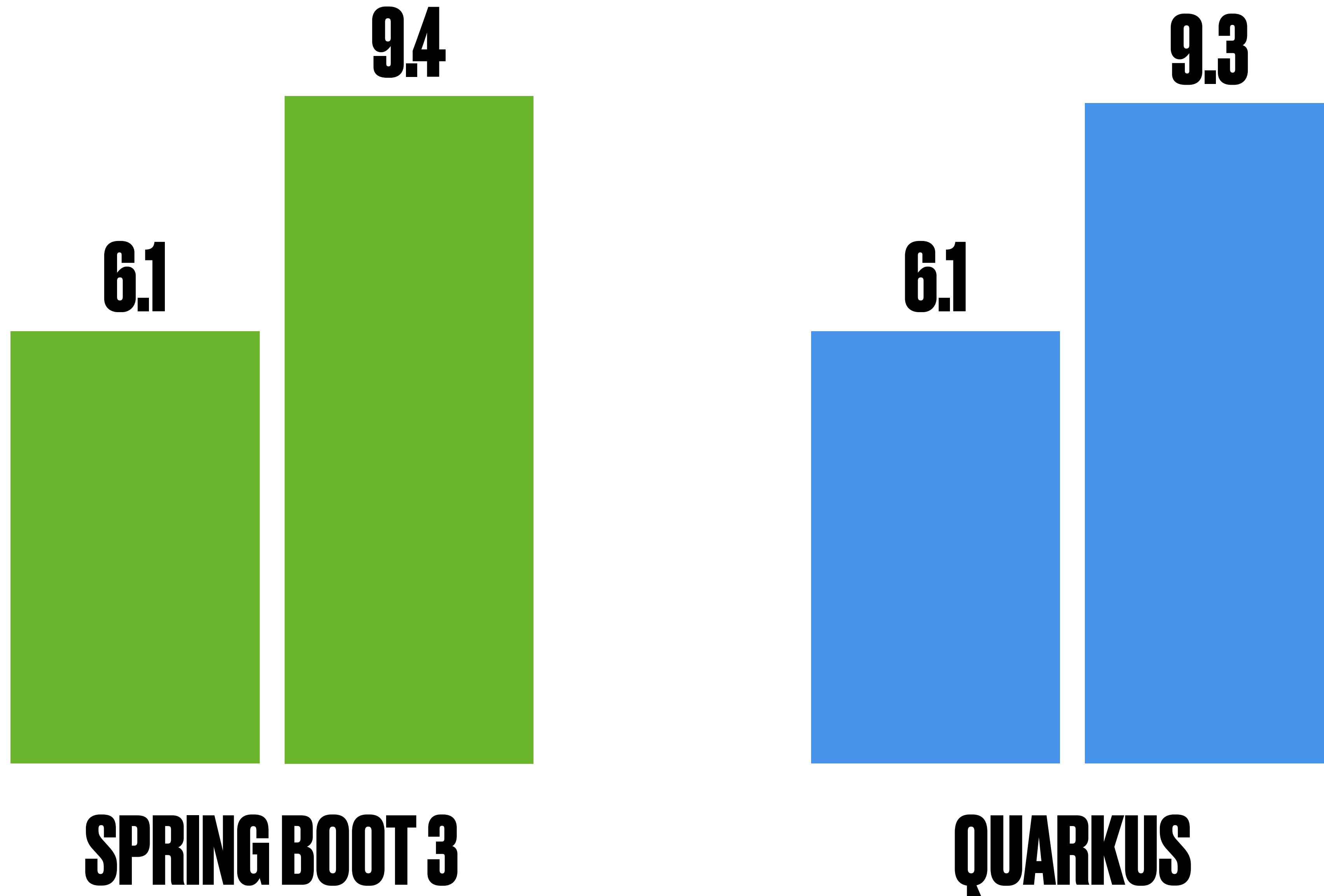
WHICH FRAMEWORK?

MY DEMO

SPRING BOOT 3 VS. QUARKUS 3 VS. GO



JIT JAVA VS. NATIVE JAVA: PROCESSING TIME



NATIVE JAVA: ONLY

65-66% OF JIT JAVA

PERFORMANCE

QUARKUS: BEST START
TIME & MEMORY USE...

...VS. FRAMEWORK
SWITCHING COSTS

MICRONAUT:

"SPRING BOOT MINUS

REFLECTION"

FRAMEWORK RECOMMENDATIONS

ALL FRAMEWORKS

TRY A DIFFERENT
FRAMEWORK - SPRING
BOOT 3, QUARKUS, OR
MICRONAUT

SPRING BOOT

FIRST MIGRATE TO 3.X

THEN TEST NATIVE
JAVA

QUARKUS & MICRONAUT

KEEP IT IF IT WORKS



1. DOES THE BOSS CARE?



2. DO THE NUMBERS ADD UP?



3. COULD IT WORK?



4. START SMALL

5. GO BIG

GO BIG

GO BIG

CAREFULLY CONTINUE TO ROLL OUT NATIVE JAVA

ADJUST CONSTANTLY

**ONLY MIGRATE APPLICATIONS TO NATIVE JAVA IF
IT ADDS UP**

- ✓ **1. DOES THE BOSS CARE?**
- ✓ **2. DO THE NUMBERS ADD UP?**
- ✓ **3. COULD IT WORK?**
- ✓ **4. START SMALL**
- ✓ **5. GO BIG**

FOR **MOST** OF YOU:
NOT NOW

WHY NOT?

- ✘ 1. DOES THE BOSS CARE?**
- ✘ 2. DO THE NUMBERS ADD UP?**
- 3. COULD IT WORK?**
- 4. START SMALL**
- 5. GO BIG**

HARD TO HIRE FOR

WHY NOW?

ADDITIONAL DEVELOPER

TIME MAY DECREASE

**LIBRARIES NOT
WORKING MAY
DECREASE**

SAVINGS MAY INCREASE

WORTH IT WHEN?

AGENDA

PROBLEM?

CDS & CRAC

GRAALVM

WORTH IT WHEN?

SUMMARY

JAVA EXPENSIVE:
TUNED FOR LONG-LIVED
BIG APPLICATIONS

**JAVA'S NEW
COMPETITION:
JAVASCRIPT & PYTHON**

JAVA CHEAPER:

GDS, GRAC¹²³,

GRAALVM⁴⁵⁶⁷⁸

FASTER START
LESS MEMORY
SMALLER FILES
BETTER SECURITY

START SB 2 – QUARKUS

TIME (MS): 589 – 10.4

RAM (MB): 107 – 7.3

MONOLITH



MICROSERVICES



SERVERLESS



**WHEN IS NATIVE JAVA
WITH GRAALVM
WORTHWHILE FOR ME?**

FOR **MOST** OF YOU:
NOT NOW



THANK
YOU

QUARKUS TEAM, RED HAT

BEN EVANS

DIMITRIS ANDREADIS

FOLVOS ZAKKAK

GALDER ZAMARRENO

HOLLY CUMMINS

MAX RYDAHL ANDERSEN

MICHAEL KAM BARBACEC

PATRICK BAUMGARTNER

SANNE GRINOVERO

OPENJ9 TEAM, RED HAT

DAN HEIDINGA

GRAALVM TEAM, ORACLE

ALINA YURENKO

AZUL

SIMON RITTER

The image features a classic 'The End' title card. It consists of a series of concentric circles in shades of red, with a solid black circle at the very center. The words 'The End' are written in a white, cursive script font across the middle of the circles. The text is slightly offset to the right, with 'The' on the left and 'End' on the right, both overlapping the black center and the red rings.

The End

MY **TALK** DOESN'T STOP
WHEN I STOP **TALKING**TM

SLIDES

ADDITIONAL INFORMATION

DEMO CODE REPOS

GET STARTED WITH NATIVE JAVA

JAVA TECH POPULARITY NEWSLETTER



BPFLI/LAB