SEARCH (/SEARCH) SITE MAP (/SITE-MAP)

MICRONAUT VS QUARKUS VS SPRING BOOT PERFORMANCE ON JDK 14

HOME (HTTPS://OBJECTCOMPUTING.COM/) > NEWS (HTTPS://OBJECTCOMPUTING.COM/NEWS) > MICRONAUT VS QUARKUS VS SPRING BOOT PERFORMANCE ON JDK 14

APRIL 07, 2020

MICRONAUT

MICRONAUT VS QUARKUS VS SPRING BOOT PERFORMANCE ON JDK 14

By Graeme Rocher, OCI Grails & Micronaut Product Lead and Principal Software Engineer

APRIL 6, 2020

CATEGORIES

News (https://objectcomputing.com/news/category/news)Grails Blog (https://objectcomputing.com/news/category/grails-blog)Micronaut Blog (https://objectcomputing.com/news/category/micronaut-blog) (https://objectcomputing.com/news/category/alyce-ai)

SUBSCRIBE

First Name

^

After the recent releases of Micronaut 2.0 M2 and JDK 14, I decided it would be a good time to see what the state of play is performance wise between Micronaut, Quarkus, and Spring Boot on JDK 14 in 2020.

Historically, we have encouraged users to make their own judgments, but recently some incorrect or distorted information has been published, and I think it's important that I clarify some points that may be confusing or misleading. For example, if you search for "Micronaut vs Quarkus" on Google, the first result it serves up is a blog post (https://simply-how.com/quarkus-vs-micronaut) that contains benchmarking data that none of the Micronaut team has been able to replicate with any recent version of Micronaut on a variety of systems.

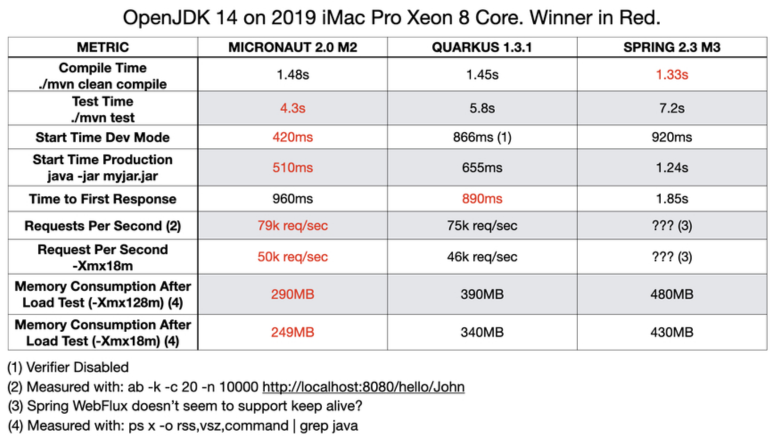
So to ensure there is more accurate information out there, and also to show users how to compare these frameworks in an "apples-to-apples" manner (since each framework's default configuration is different), I decided to take for a spin the latest versions of the three most-hyped frameworks among Java developers today: Micronaut, Quarkus and Spring Boot.

Before getting started, we acknowledged that this cannot help but be a somewhat biased analysis, given that Object Computing is the home of Micronaut. To mitigate this bias, we attempted to make the comparison as transparent as possible by

TAGS

Alyce.ai (https://objectcomputi
AngularJS (https://objectcomp
AWS (https://objectcomputing.
Blockchain (https://objectcomp
Case Study (https://objectcom
Clients (https://objectc

^


OpenJDK 14 on 2019 iMac Pro Xeon 8 Core. Winner in Red.

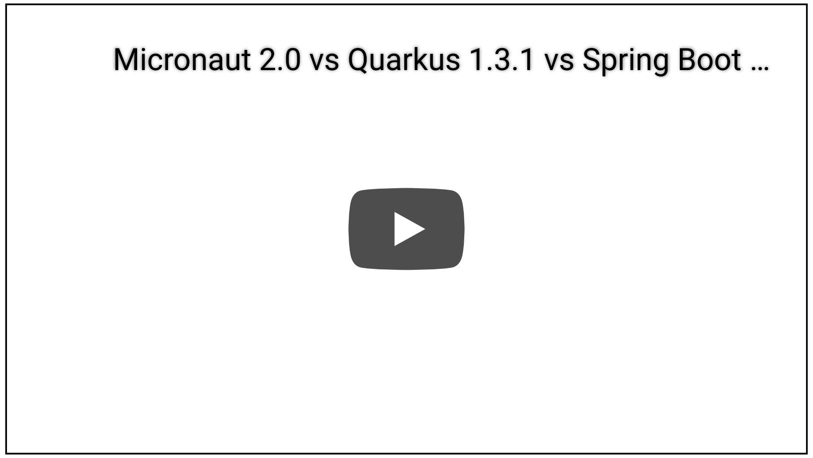
METRIC	MICRONAUT 2.0 M2	QUARKUS 1.3.1	SPRING 2.3 M3
Compile Time ./mvn clean compile	1.48s	1.45s	1.33s
Test Time ./mvn test	4.3s	5.8s	7.2s
Start Time Dev Mode	420ms	866ms (1)	920ms
Start Time Production java -jar myjar.jar	510ms	655ms	1.24s
Time to First Response	960ms	890ms	1.85s
Requests Per Second (2)	79k req/sec	75k req/sec	??? (3)
Request Per Second -Xmx18m	50k req/sec	46k req/sec	??? (3)
Memory Consumption After Load Test (-Xmx128m) (4)	290MB	390MB	480MB
Memory Consumption After Load Test (-Xmx18m) (4)	249MB	340MB	430MB

(1) Verifier Disabled
(2) Measured with: ab -k -c 20 -n 10000 http://localhost:8080/hello/John
(3) Spring WebFlux doesn't seem to support keep alive?
(4) Measured with: ps x -o rss,vsz,command | grep java

UPDATE: The response from the Spring team was to use wrk (https://github.com/wg/wrk) or vegata (https://github.com/tsenart/vegeta) to benchmark instead. Quarkus appears to be the winner at higher concurrency when testing with wrk whilst Micronaut appears to be the winner with vegata. This proves once again do your own benchmarking with a variety of tools!

As you can see, as of today, if you are looking at performance, Quarkus is marginally ahead on time to first response (around 70ms), while the only metric Spring Boot wins is compilation time due to not doing any compilation-time processing.

With regard to Spring Boot, we were unable to extract reliable request-per-second data because Spring's Netty implementation bizarrely doesn't seem to performing tests that users can replicate at home, and to conduct them live in a completely unedited screencast. You can watch that screencast here:



For reference, the source code for the examples built during the recording of this screencast can be found on Github (https://github.com/graemerocher/framework-comparison-2020).

SUMMARY OF RESULTS

Following are the stats that were produced during the recording of this screencast for the different metrics we tested (winner in red), taking the averages over 10 runs:

Cloud Engineering (https://obje
Community Outreach (https://c
Data (https://objectcomputing.
DDS (https://objectcomputing.
Defense (https://objectcomput
DevOps (https://objectcomputi
Events (https://objectcomputi
Finance (https://objectcomputi
Google Cloud Platform (https:/
GORM (https://objectcomputin
Gradle (https://objectcomputin
Grails (https://objectcomputin
Groovy (https://objectcomputir
Healthcare (https://objectcomp
IIoT (https://objectcomputing.
Java (https://objectcomputing.
JavaScript (https://objectcomp
JVM (https://objectcomputing.
Machine Learning (https://obje
Manufacturing (https://objectc
Micronaut (https://objectcomp
Microservices (https://objectc
Middleware (https://objectcom
Open Source (https://objectco
OpenDDS (https://objectcompu
React (https://objectcomputing
Security (https://objectcomput
Software Engineering (https://
Spring Boot (https://objectcom
Training (https://object
Web Development (http

support keep-alive connections correctly. Of course this may be something that we are doing wrong on our side, so please feel free to leave a comment on the video if any Spring users know the solution.

The Quarkus team has made bold claims about the memory efficiency of Quarkus, so it was surprising to see such a disparity when actual tests were conducted that seem to disprove these claims. The Micronaut team and I are disappointed that we had to take it upon ourselves to perform these tests and publish the results, not as a simple opportunity to help others improve their processes and applications, but to respond to misinformation that could, theoretically, do the opposite.

We are all extremely proud of Micronaut, and we know users have been happy with its powerful capabilities. We listen to feedback and work every day on making improvements that we hope will help even more people enhance their productivity and build faster, better apps. That said, I encourage all users to do their own testing and make their own judgments regarding which JVM framework they prefer.

GET GROOVY, GRAILS, AND MICRONAUT UPDATES IN YOUR INBOX

First Name*

^

Last Name*

Email*

Your info is safe with us; check out our privacy policy (https://objectcomputing.com/about/legal/privacy-statement) for details.*

○ Got it. Let's proceed.

KEEP ME IN THE LOOP!

View our privacy policy ({CCM:BASE_URL}/about/legal/privacy-statement)

← Micronaut 2.0 Milestone 2 - Massive Maven Improvements (https://objectcomputing.com/news/2020/04/02/micronaut-20-milestone-2-massive-maven-improvements)

Disruptive Innovation Found In Unexpected Places (https://objectcomputing.com/news/2020/04/08/innovation-found-unexpected-places)

^

NEWS ∨

EVENTS ∨

CONTACT ∨

^