SpringOne TOUR

by VMware Tanzu

Scaling Your Spring Boot App to Zero

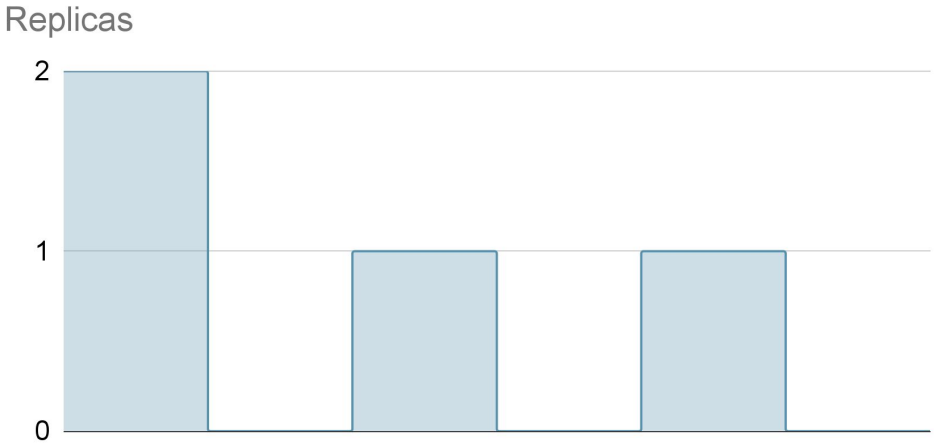
DaShaun Carter
Spring Developer Advocate, VMware

Thomas Risberg
Staff Engineer, VMware

https://twitter.com/kelseyhightower/status/1118181786914136064

Scale to zero

- Generalization of serverless
- Typically hosted on Kubernetes platforms based on Knative/KEDA
- Not limited to functions
- Pay-as-you-use billing model



spring

2 ways to scale your Spring applications to zero with sub-second startup

spring

1 Scale to zero with native Spring Boot and GraalVM

Spring AOT
GraalVM Native Images

A new deployment variant

As an alternative to classic HotSpot deployment

spring

```
$ ./gradlew bootBuildImage
Successfully built image 'docker.io/library/spring-petclinic:1.0.0-SNAPSHOT'

$ docker run --rm -p 8080:8080 spring-petclinic:1.0.0-SNAPSHOT
              |\      _,,,--,,_
             /,`.-'`'   ._  \-;;,_
  _______ __|,4-  ) )_   .;.(__`'-'__     ___ __    _ ___ _______
 |       | '---''(_/._)-'(_\_)   |   |   |   |  |  | |   |       |
 |    _  |    ___|_     _|       |   |   |   |   |_| |   |       | __ _ _
 |   |_| |   |___  |   | |       |   |   |   |       |   |       | \ \ \ \
 |    ___|    ___| |   | |      _|   |___|   |  _    |   |      _|  \ \ \ \
 |   |   |   |___  |   | |     |_|       |   | | |   |   |     |_    ) ) ) )
 |___|   |_______| |___| |_______|_______|___|_|  |__|___|_______|  / / / /
 ==================================================================/_/_/_/

:: Built with Spring Boot :: 3.1.4


Starting AOT-processed PetClinicApplication using Java 17.0.8.1 with PID 1
No active profile set, falling back to 1 default profile: "default"
Tomcat initialized with port(s): 8080 (http)
Starting service [Tomcat]
Starting Servlet engine: [Apache Tomcat/10.1.13]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 19 ms
HikariPool-1 - Starting...
HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=PETCLINIC
HikariPool-1 - Start completed.
Exposing 13 endpoint(s) beneath base path '/actuator'
Tomcat started on port(s): 8080 (http) with context path ''
Started PetClinicApplication in 0.107 seconds (process running for 0.109)
```

Sub-second startup on production!

GraalVM advantages

Instant startup

Milliseconds for native instead of seconds for the JVM

No warmup

Peak performance available immediately

Low resource usage

Lower memory footprint and no JIT compilation

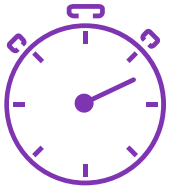
Reduced surface attack

Closed world of dependencies with explicit reflection and serialization

Compact packaging

Smaller container easier to deploy

GraalVM trade-offs



Very slow compilation

Minutes instead of seconds

Compatibility

Reachability metadata required for reflection, proxies, resources

Closed-world Assumptions

Bean conditions fixed at build time
No dynamic class loading

Thomas Wuerthinger
@thomaswue

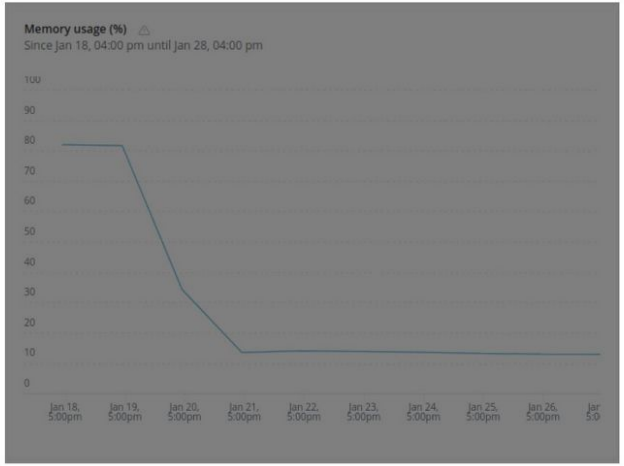
"The startup time of the application was reduced from approximately 30 seconds down to about 3 ms, and more importantly the memory usage was also significantly reduced from 6.6 GB down to 1 GB, with the same throughput and CPU utilization." 👍🚀 @graalvm

> **Thomas Schuehly** @tschuehly · 21 Jul
> 10minutemail.com runs on @GraalVM @springboot native image with server-side rendering with @thymeleaf 🤯
> digitalsanctuary.com/10minutemail/m...

https://twitter.com/thomaswue/status/1682465475748298755

Thomas Schuehly
@tschuehly

10minutemail.com runs on @GraalVM @springboot native image with server-side rendering with @thymeleaf 🤯
digitalsanctuary.com/10minutemail/m...



Migrating 10MinuteMail from Java to GraalVM Native

12:59 pm · 21 Jul 2023 · **36K** Views

16 Retweets **2** Quotes **88** Likes **29** Bookmarks

spring

Thomas Wuerthinger
@thomaswue

Big news from last week's @GraalVM for @Java 21 release was that we can finally demonstrate that AOT (with profile-guided optimizations) can outperform JIT for all major metrics including throughput! Here are the numbers for the @Springframework PetClinic example (both G1 GC). 🚀

	GraalVM CE with C2 JIT	Oracle GraalVM Native Image	
Memory Usage (max RSS)	1,029 MB	641 MB	-38% lower
Peak throughput	11,066 req/s	11,902 req/s	+8% higher
Throughput per memory	12,488 req/(GB*s)	18,569 req/(GB*s)	+49% better
Tail latency (P99)	7.2ms	5.15ms	-28% lower
Startup	7,090ms	210ms	34x faster

11:44 AM · Sep 29, 2023 · **25.3K** Views

https://twitter.com/thomaswue/status/1707783370187440148

spring

Demo - *Spring PetClinic native build running as Knative service*

2 Scale to zero with the JVM Spring Boot and Project CRaC

spring

JVM Checkpoint Restore

Project CRaC

Immediate startup for Spring deployments on HotSpot

spring

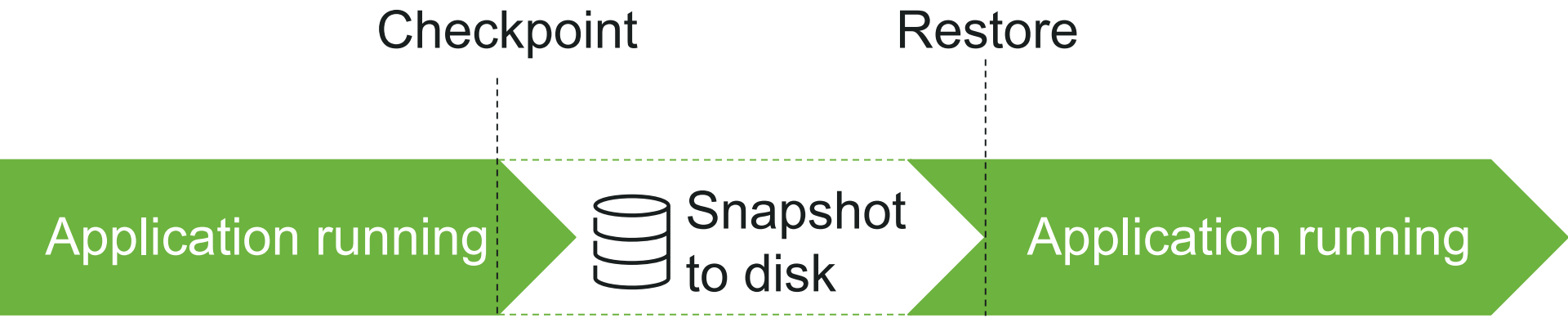
Coming up...

Spring Framework 6.1

Spring Boot 3.2

November 2023

JVM Checkpoint Restore



spring

Spring Boot 3.2 introduces initial support for JVM Checkpoint Restore with Project CRaC

spring

Check the evolving scope of CRaC support

https://github.com/spring-projects/spring-checkpoint-restore-smoke-tests/blob/main/STATUS.adoc

Boot

Smoke Test	appTest	checkpointRestoreAppTest	test
actuator-webflux	build passing	build passing	
actuator-webmvc	build passing	build passing	

Cloud

Smoke Test	appTest	checkpointRestoreAppTest	test
context-refresh	build passing	build passing	build passing
context-refresh-http	build passing	build passing	build passing

Data

Smoke Test	appTest	checkpointRestoreAppTest	test
data-jdbc	build passing	build passing	
data-jpa	build passing	build passing	
data-redis	build passing	build passing	

Framework

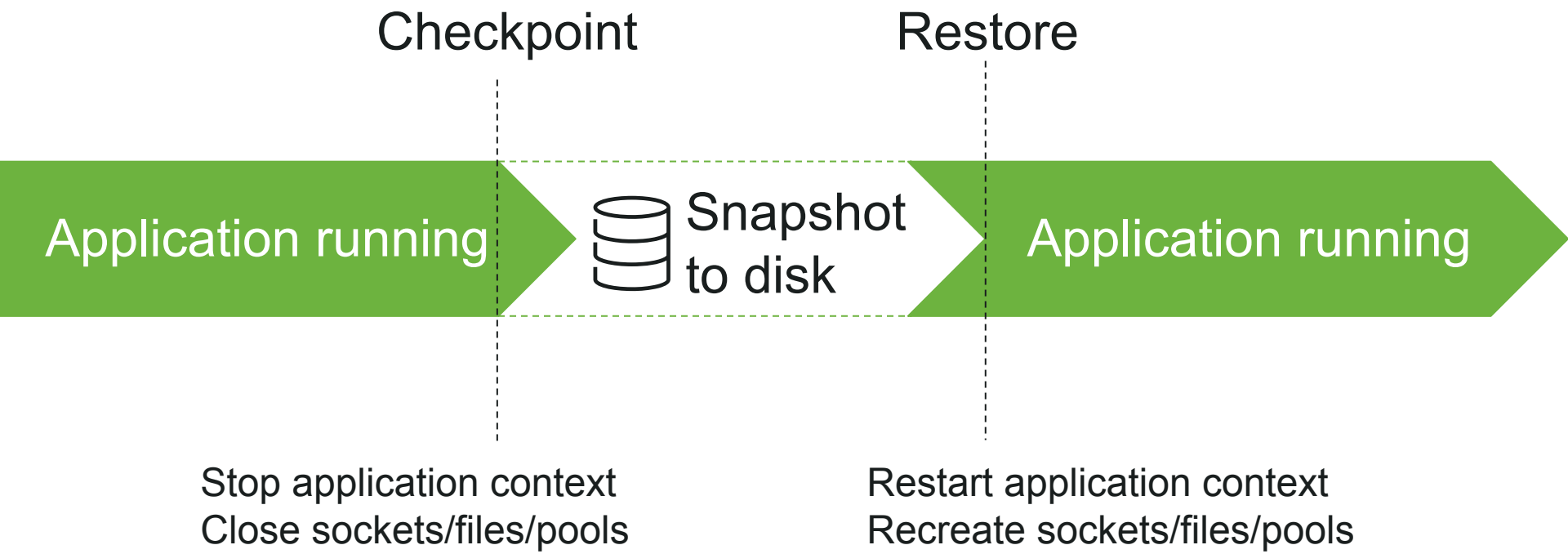
Smoke Test	appTest	checkpointRestoreAppTest	test
hibernate-mysql	build passing	build passing	
scheduled	build passing	build passing	
webclient-netty	build passing	build passing	
webflux-netty	build passing	build passing	build passing
webflux-undertow	build passing	build passing	build passing
webmvc-jetty	build passing	build passing	build passing
webmvc-tomcat	build passing	build passing	build passing

Integration

Smoke Test	appTest	checkpointRestoreAppTest	test
integration-basic	build passing	build passing	
integration-webfux-data	build passing	build passing	
spring-amqp-rabbit	build passing	build passing	
spring-kafka	build passing	build passing	
spring-kafka-avro	build passing	build passing	
spring-kafka-streams	build passing	build passing	

spring

Spring lifecycle with CRaC



spring

```
              |\      _,,,--,,_
             /,`.-'`'    ._  \-;;,_
  _______ __|,4-  ) )_   .;.(__`'-'__     ___ __    _ ___ _______
 |       | '---''(_/._)-'(_\_)   |   |   |   |  |  | |   |       |
 |    _  |    ___|_     _|       |   |   |   |   |_| |   |       | __ _ _
 |   |_| |   |___  |   | |       |   |   |   |       |   |       | \ \ \ \
 |    ___|    ___| |   | |      _|   |___|   |  _    |   |      _|  \ \ \ \
 |   |   |   |___  |   | |     |_|       |   | | |   |   |     |_    ) ) ) )
 |___|   |_______| |___| |_______|_______|___|_|  |__|___|_______|  / / / /
 ==================================================================/_/_/_/

:: Built with Spring Boot :: 3.2.0-M3


Starting PetClinicApplication v0.0.1-SNAPSHOT using Java 17.0.8.1 with PID 129
No active profile set, falling back to 1 default profile: "default"
Bootstrapping Spring Data JDBC repositories in DEFAULT mode.
Finished Spring Data repository scanning in 45 ms. Found 5 JDBC repository interfaces.
Tomcat initialized with port 8080 (http)
Starting service [Tomcat]
Starting Servlet engine: [Apache Tomcat/10.1.13]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 1443 ms
HikariPool-1 - Starting...
HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=PETCLINIC
HikariPool-1 - Start completed.
Exposing 13 endpoint(s) beneath base path '/actuator'
Tomcat started on port 8080 (http) with context path ''
Started PetClinicApplication in 3.102 seconds (process running for 3.467)
Evicting Hikari connections
CR: Checkpoint ...



Restarting Spring-managed lifecycle beans after JVM restore
Tomcat started on port 8080 (http) with context path ''
Spring-managed lifecycle restart completed in 8 ms (restored JVM running for 50 ms)
```

Regular startup typically on CI/CD platform

Optional load testing to warm up the JVM then checkpoint

Sub-second restoration on production with the JVM!

Project CRaC advantages

Instant startup

Milliseconds instead of seconds for the regular JVM

No runtime warmup

Peak performance available immediately

Compatibility

Still a JVM with all regular capabilities

Project CRaC trade-offs

Checkpoint startup

Require to start the application ahead

Lifecycle management

Require to close and reopen sockets, files, pools

Secret management

Sensitive informations may leak in the snapshot files

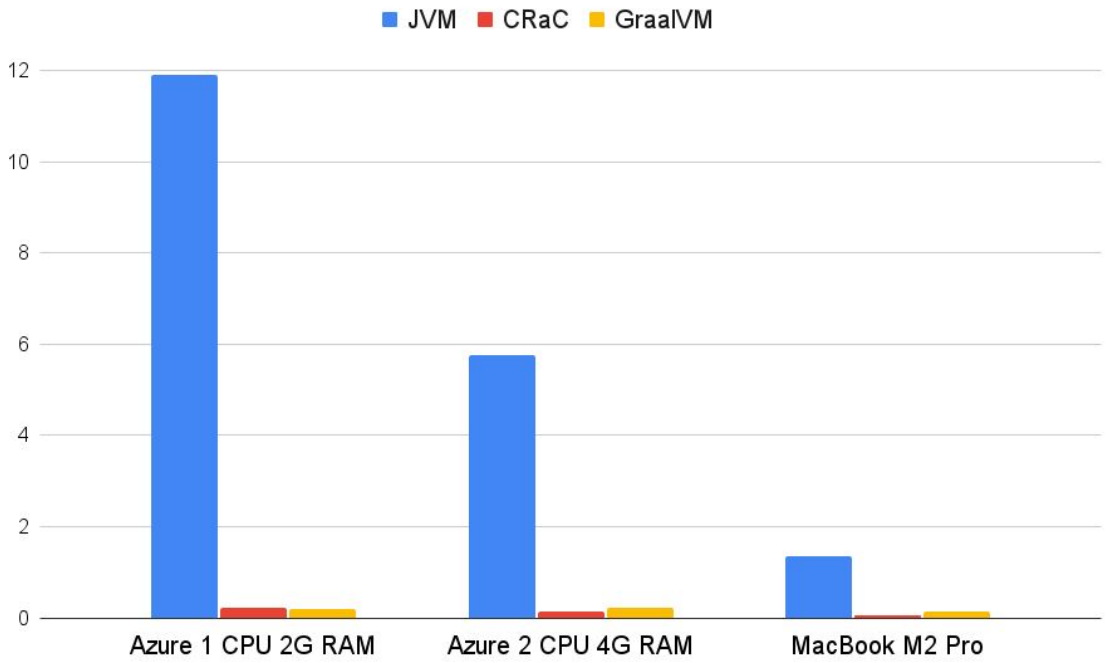
System Integration

Linux only and advanced capability management required

Demo - *Getting started with JVM Checkpoint/Restore builds*

Petclinic startup time (seconds)

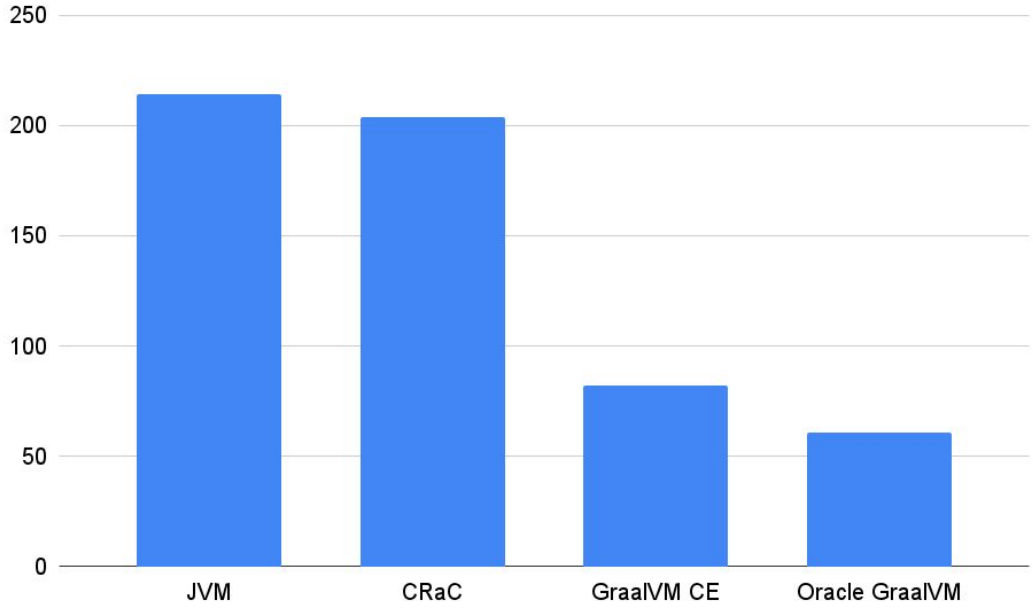
GraalVM and CRaC both allow 50x faster startup time and scaling to zero



spring

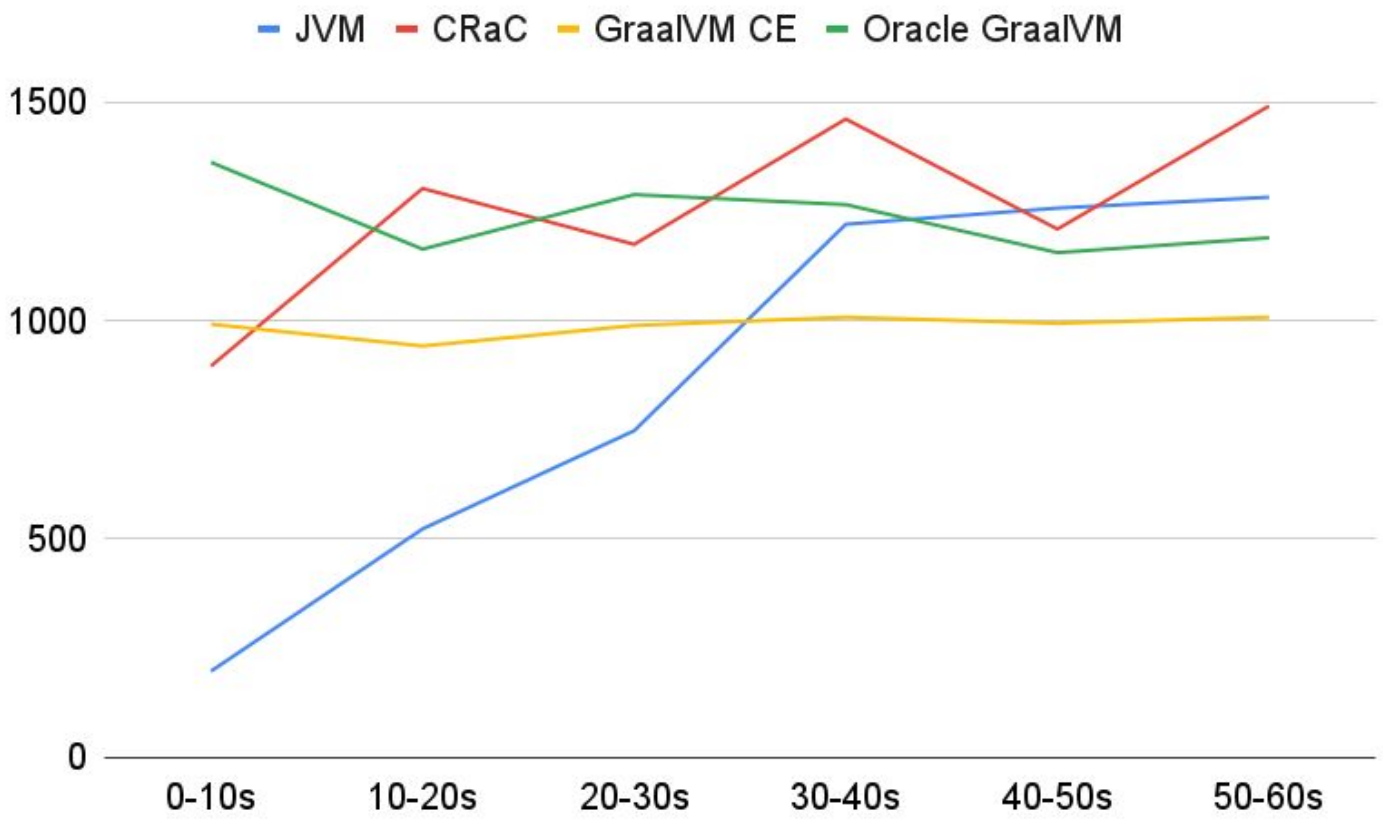
Petclinic memory consumption after startup (MB)

GraalVM allows 3x reduction of memory consumption of Spring Boot application infrastructure



spring

Petclinic throughput on 1 CPU 2G RAM (req/s)



spring

Thank you

Follow us
@dashaun
@trisberg

Dig Deeper:
https://github.com/trisberg/springone-tour-2023

SpringOne TOUR
by VMware Tanzu