# Supercharge Your JVM Performance with Project Leyden & Spring Boot

**Ana-Maria Mihalceanu**
Senior Developer Advocate
Java Platform Group @ Oracle

**Moritz Halbritter**
Spring Engineering
Team Member @ Broadcom

# Goals

Understand how Project Leyden boosts Java startup and performance.

Demonstrate how to use Leyden-related optimizations available in JDK 25 with Spring Boot.

Learn techniques you can apply today along with insights into what’s ahead.


# Performance is a journey, not a finish line.

# Project Leyden

Improve the startup time
and warmup time
of Java applications

by shifting computation
earlier or later in time.

## Static Analysis

## Dynamic Observation

# Training Runs

**Purpose**

Exercise the startup and warmup code paths, under observation.

Discover ahead-of-time what you'd otherwise find early at runtime.

**What Constitutes a Training?**

**Best training run is observing the application in production.**

Usually needs writing a small driver program (like an integration test).

Runs at build time (similar to an integration test).

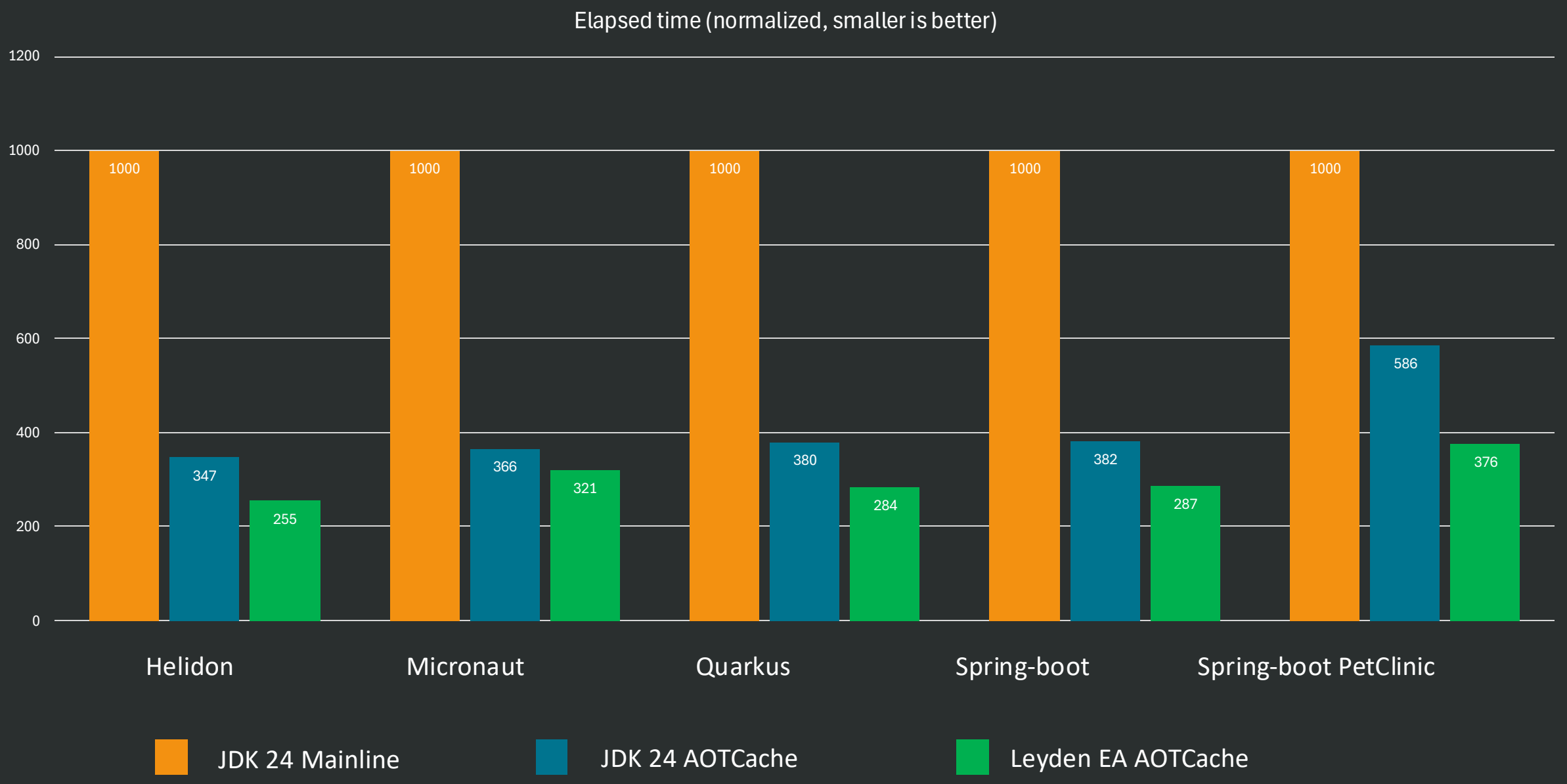
**Why They Work?**

Effective for the same reason dynamic compilation is.

Analysis is driven by what the program actually does.

# JDK 24’s Three-Phase Workflow

```
$ # Training Run
$ java -XX:AOTMode=record -XX:AOTConfiguration=app.aotconf \
       -cp app.jar com.example.App ...

$ # Assembly Phase
$ java -XX:AOTMode=create -XX:AOTConfiguration=app.aotconf \
       -XX:AOTCache=app.aot -cp app.jar

$ # Deployment Run
$ java -XX:AOTCache=app.aot -cp app.jar com.example.App ...
```




Source: https://github.com/openjdk/leyden/blob/634547513c2a2b707ae43a735dc24fd1977da2ae/README.md


## JEP 483: Ahead-of-Time Class Loading & Linking

| | |
|---|---|
| *Authors* | Ioi Lam, Dan Heidinga, & John Rose |
| *Owner* | Ioi Lam |
| *Type* | Feature |
| *Scope* | JDK |
| *Status* | Closed / Delivered |
| *Release* | 24 |
| *Component* | hotspot / runtime |
| *Discussion* | leyden dash dev at openjdk dot org |
| *Reviewed by* | Alex Buckley, Brian Goetz, Mark Reinhold, Vladimir Kozlov |
| *Endorsed by* | Vladimir Kozlov |

## JEP 515: Ahead-of-Time Method Profiling

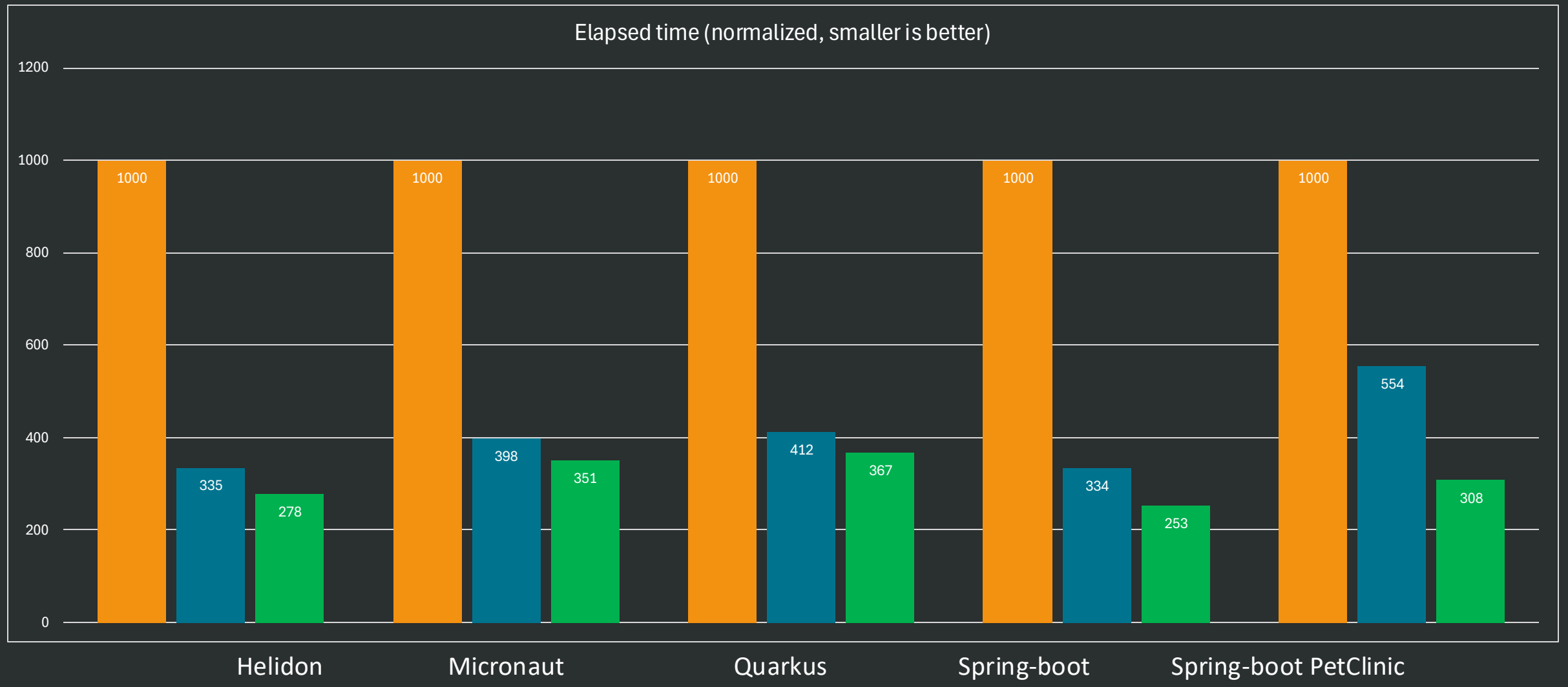
| | |
|---|---|
| *Author* | Igor Veresov & John Rose |
| *Owner* | John Rose |
| *Type* | Feature |
| *Scope* | Implementation |
| *Status* | Proposed to Target |
| *Release* | 25 |
| *Component* | hotspot / compiler |
| *Discussion* | leyden dash dev at openjdk dot org |
| *Effort* | M |

## JEP 514: Ahead-of-Time Command-Line Ergonomics

| | |
|---|---|
| *Owner* | John Rose |
| *Type* | Feature |
| *Scope* | JDK |
| *Status* | Closed / Delivered |
| *Release* | 25 |
| *Component* | hotspot / runtime |
| *Discussion* | leyden dash dev at openjdk dot org |
| *Effort* | M |
| *Duration* | S |
| *Relates to* | JEP 483: Ahead-of-Time Class Loading & Linking |

# JDK 25’s Two-Phase Workflow

```
$ # Training Run + Assembly Phase
$ java -XX:AOTCacheOutput=app.aot \
       -cp app.jar com.example.App ...

$ # Deployment Run
$ java -XX:AOTCache=app.aot -cp app.jar com.example.App ...
```




Source: https://github.com/openjdk/leyden/blob/premain/README.md

# Leyden Is Fully Compatible


# AOT Cache with Spring Boot

```
erFactories    : Hibernate is in classpath; If applicable, HQL parser wil
l be used.
2025-10-01T16:05:16.462+02:00  INFO 118360 --- [           main] [
                                           ] o.s.b.a.e.web.EndpointLinks
Resolver       : Exposing 13 endpoints beneath base path '/actuator'
2025-10-01T16:05:16.511+02:00  INFO 118360 --- [           main] [
                                           ] o.s.boot.tomcat.TomcatWebSe
rver           : Tomcat started on port 8080 (http) with context path '/'
2025-10-01T16:05:16.516+02:00  INFO 118360 --- [           main] [
                                           ] o.s.s.petclinic.PetClinicAp
plication      : Started PetClinicApplication in 3.033 seconds (process r
unning for 3.319)
```

# AOT Cache Requirements

**JVM**

The same JVM must be used.

**Classpath**

Must be specified as a list of jars.

No directories, no wildcards, no nested jar.

Deployment classpath must be a superset of the training one.

**Files**

The timestamp of the jars must be preserved.

# Extract Uber Jar

```
> java -Djarmode=tools -jar application.jar extract --destination extracted

> tree extracted
extracted
├── application.jar
└── lib
    ├── commons-logging-1.3.5.jar
    ├── jackson-annotations-2.20.jar
    ├── jackson-core-3.0.0-rc9.jar
    ├── jackson-databind-3.0.0-rc9.jar
    ├── jakarta.annotation-api-3.0.0.jar
    ├── jspecify-1.0.0.jar
    ├── jul-to-slf4j-2.0.17.jar
    ├── ...
```




```
erFactories    : Hibernate is in classpath; If applicable, HQL parser wil
l be used.
2025-10-01T16:07:32.932+02:00  INFO 119101 --- [           main] [
                                           ] o.s.b.a.e.web.EndpointLinks
Resolver       : Exposing 13 endpoints beneath base path '/actuator'
2025-10-01T16:07:32.971+02:00  INFO 119101 --- [           main] [
                                           ] o.s.boot.tomcat.TomcatWebSe
rver           : Tomcat started on port 8080 (http) with context path '/'
2025-10-01T16:07:32.977+02:00  INFO 119101 --- [           main] [
                                           ] o.s.s.petclinic.PetClinicAp
plication      : Started PetClinicApplication in 2.461 seconds (process r
unning for 2.632)
```

# Ways to Do Training Runs

## Run integration tests / mirror production traffic

- Exercises many/all hot code paths used in production
- Can’t easily be done while building the image
- Collects profiling information
- More involved setup

## Stop the application before context refresh

- Easy setup
- Can be done while building the image
- Caches mostly class loading only
- Doesn’t collect profiling information
- Doesn’t exercise hot code paths in production

# Exit the Application Before Context Refresh

```
> java -Dspring.context.exit=onRefresh -jar extracted/application.jar
```

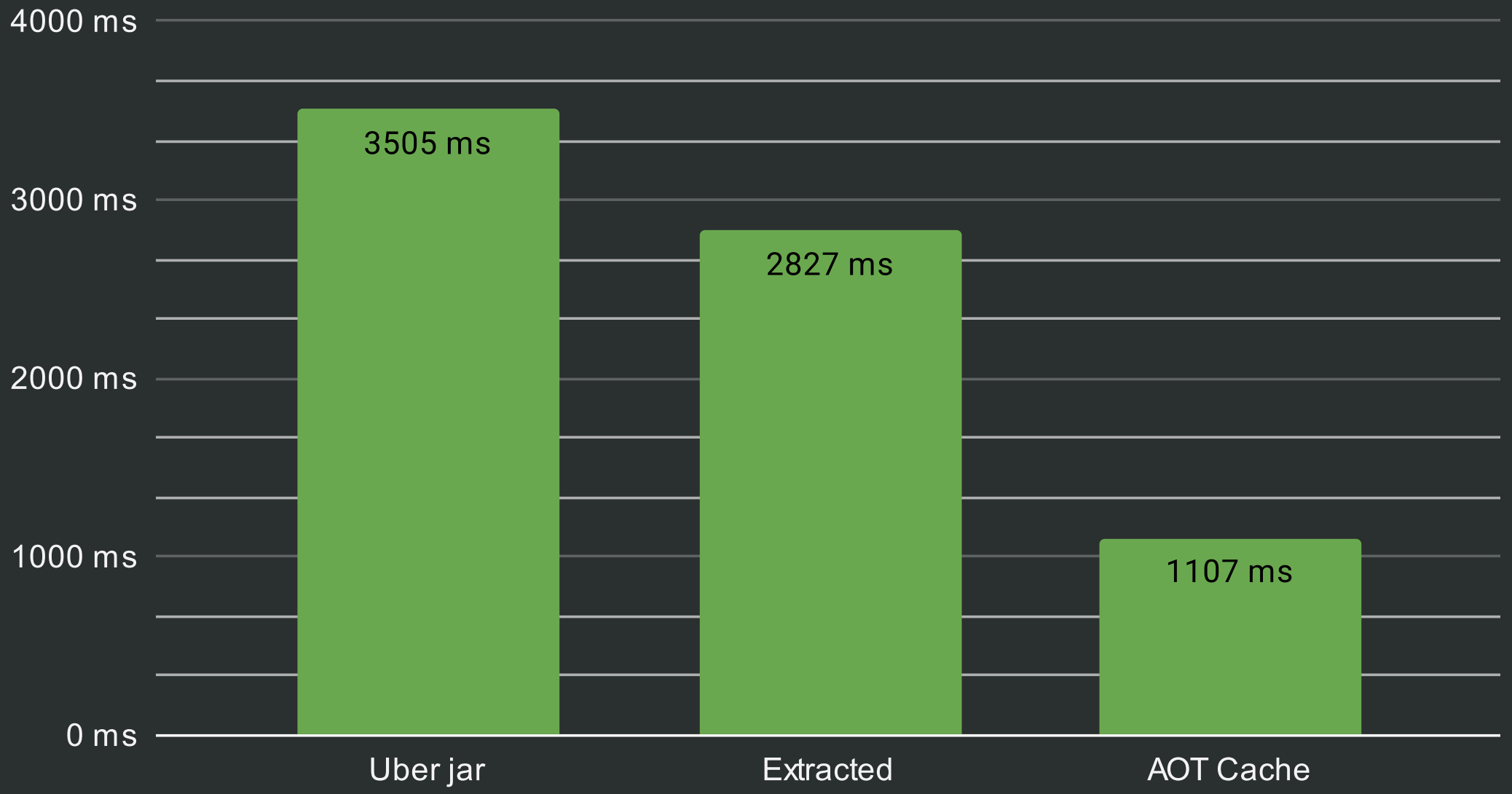
# Create and Use the AOT Cache

```
# Create the AOT cache
> java -XX:AOTCacheOutput=app.aot -Dspring.context.exit=onRefresh -jar extracted/application.jar

# Let’s see how big it is
> ll app.aot
Permissions Size Name
.rw-r--r--@  53M app.aot

# Run the application with the AOT cache
> java -XX:AOTCache=app.aot -jar extracted/application.jar
```



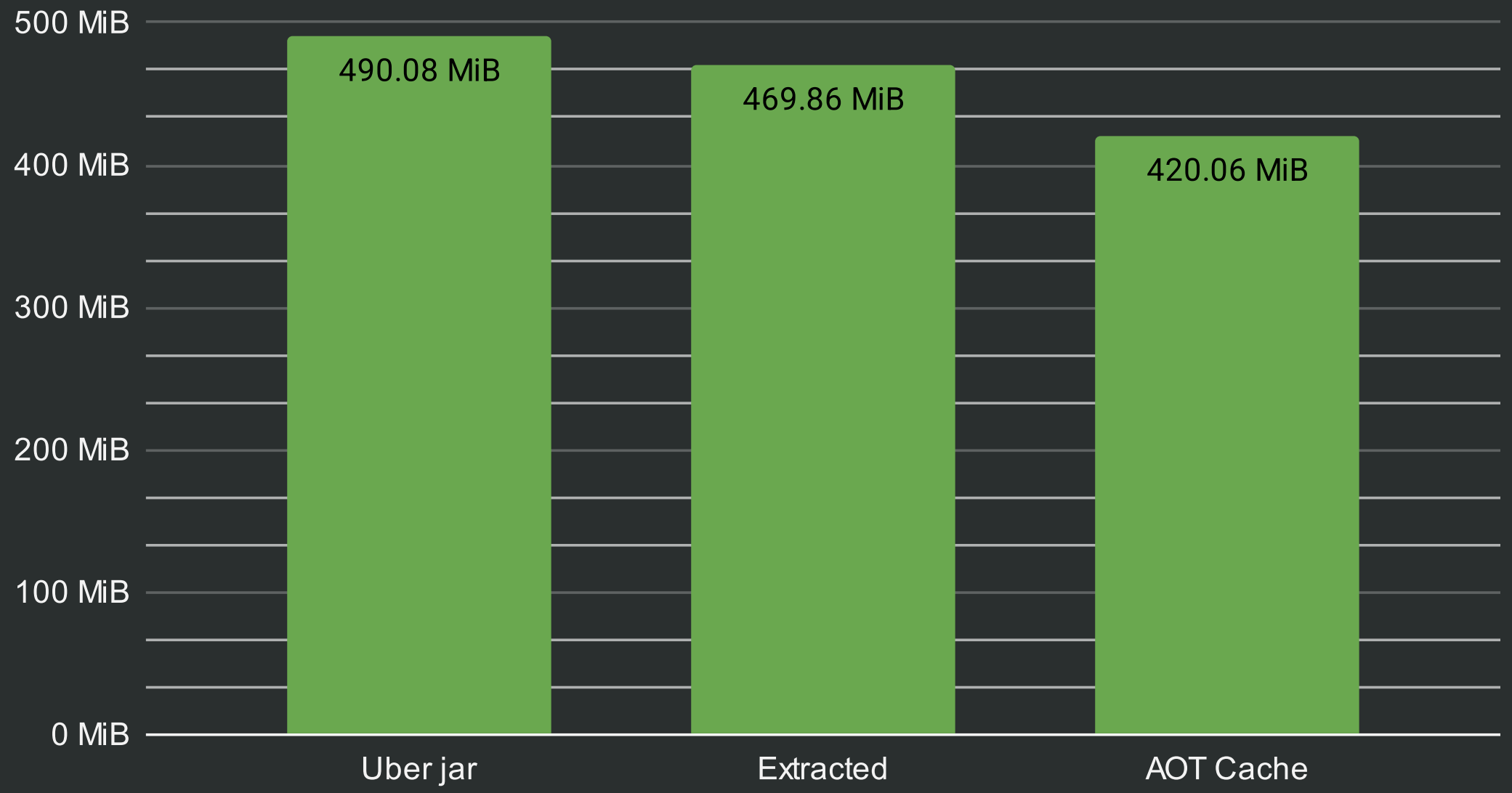

```
erFactories    : Hibernate is in classpath; If applicable, HQL parser wil
l be used.
2025-10-01T16:09:26.543+02:00  INFO 119809 --- [           main] [
                                           ] o.s.b.a.e.web.EndpointLinks
Resolver       : Exposing 13 endpoints beneath base path '/actuator'
2025-10-01T16:09:26.560+02:00  INFO 119809 --- [           main] [
                                           ] o.s.boot.tomcat.TomcatWebSe
rver           : Tomcat started on port 8080 (http) with context path '/'
2025-10-01T16:09:26.563+02:00  INFO 119809 --- [           main] [
                                           ] o.s.s.petclinic.PetClinicAp
plication      : Started PetClinicApplication in 0.839 seconds (process r
unning for 1.035)
```

# Time to first request (Spring Boot 4.0.0-M3, Java 25)

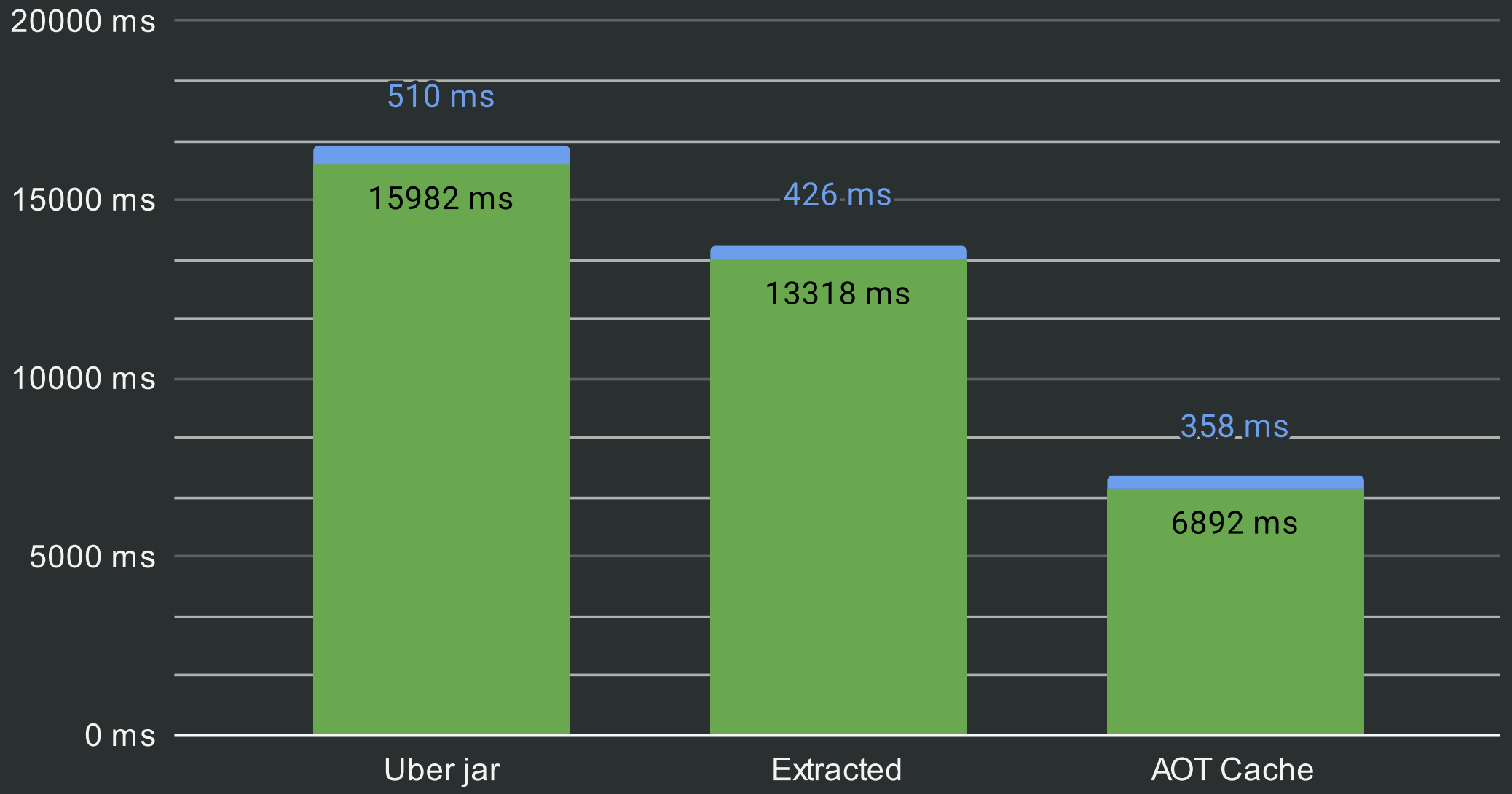



All benchmarks are using Spring Petclinic

# RSS (Spring Boot 4.0.0-M3, Java 25)



# User time / Kernel time (Spring Boot 4.0.0-M3, Java 25)



# AOT Cache and Spring AOT

## AOT Cache

JVM feature developed via Project Leyden to improve the efficiency of the JVM. It supersedes CDS.

## Spring AOT

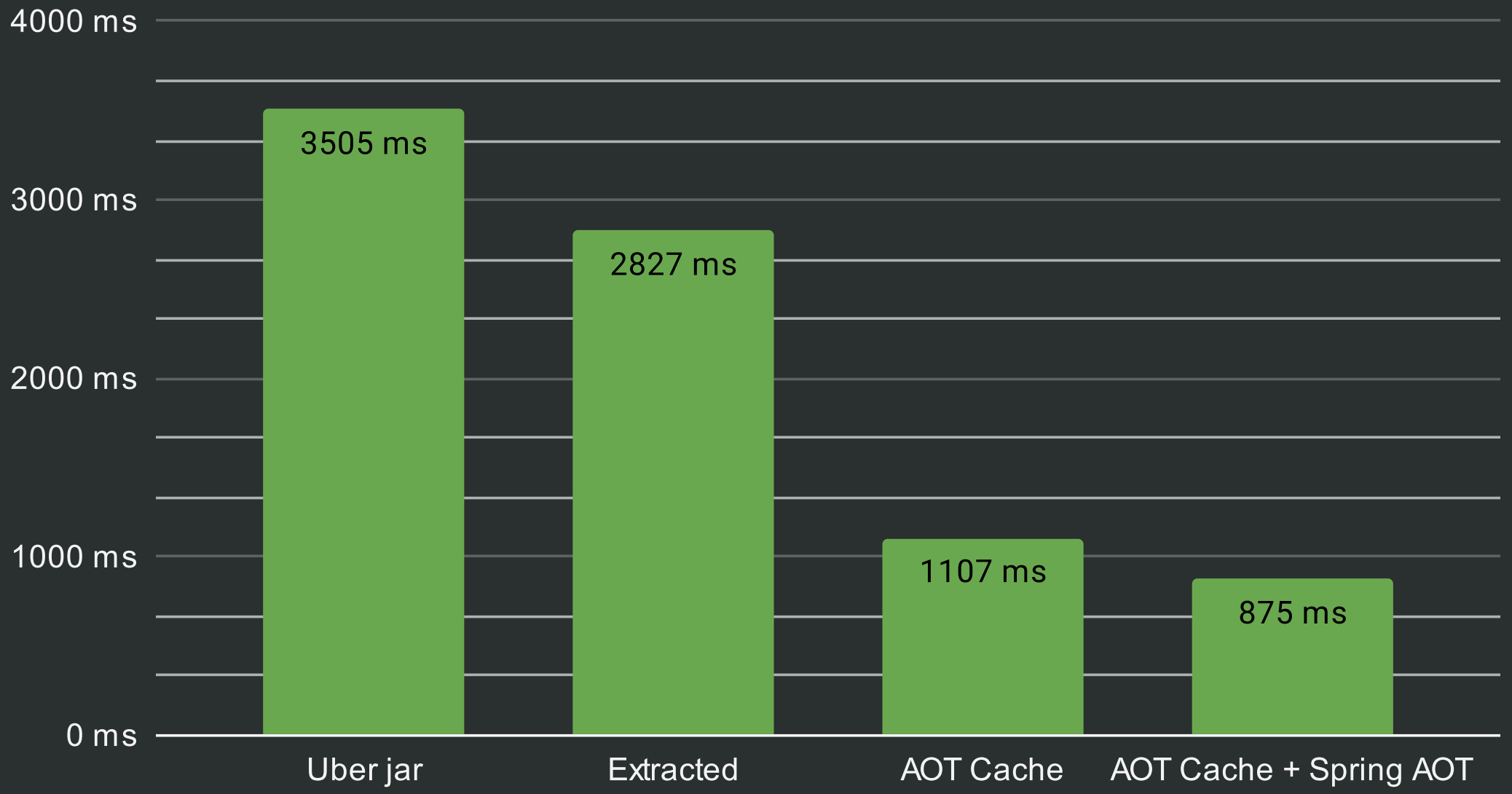
Spring feature mandatory for GraalVM native images support.

Can also be used on the JVM to speed up the startup process and lower the memory consumption.

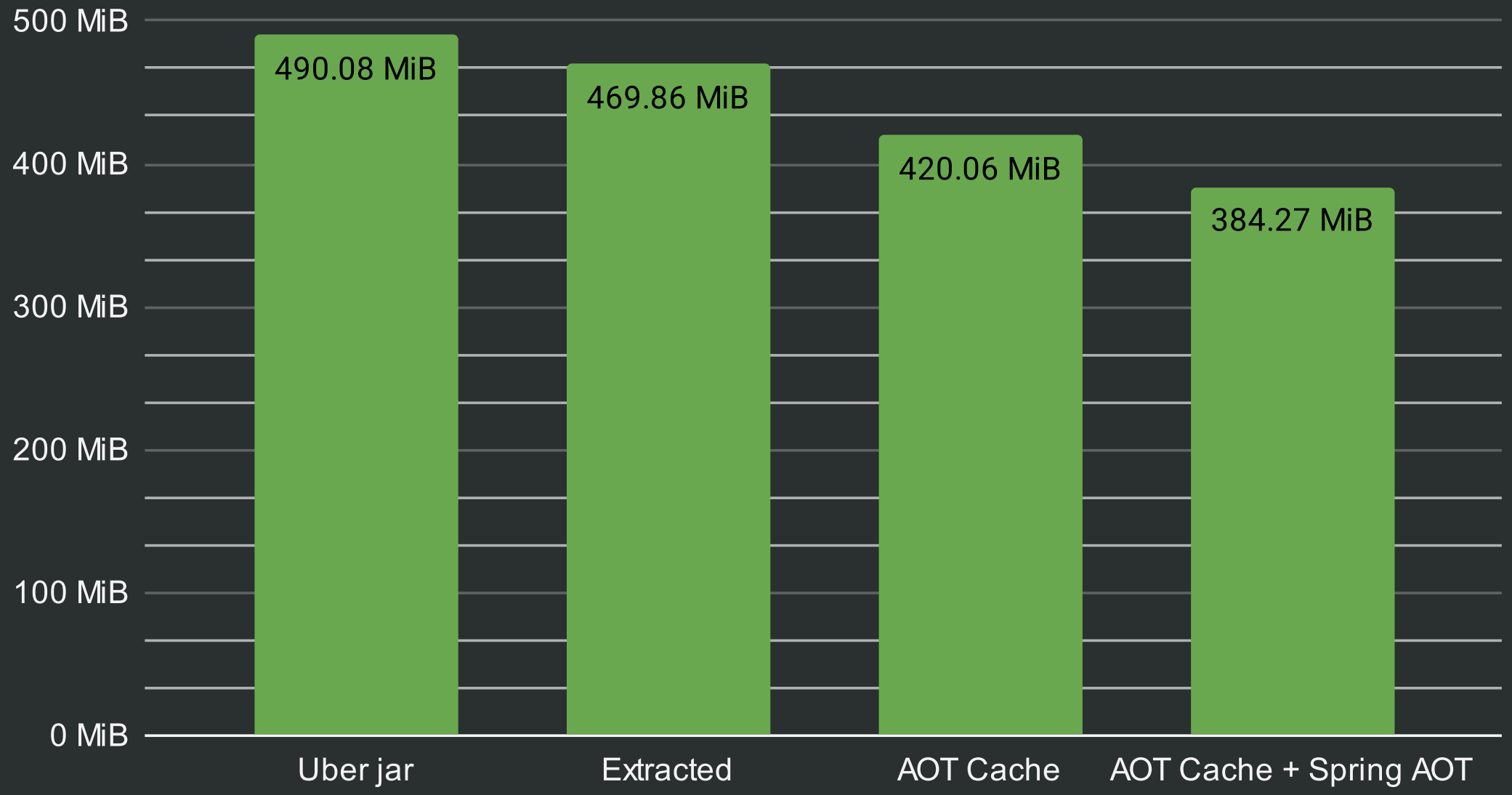
Generates code ahead of time for the bean arrangement and other features, e.g. Spring Data repositories

```
erFactories    : Hibernate is in classpath; If applicable, HQL parser wil
l be used.
2025-10-01T16:20:32.433+02:00  INFO 124278 --- [           main] [
                                          ] o.s.b.a.e.web.EndpointLinks
Resolver       : Exposing 13 endpoints beneath base path '/actuator'
2025-10-01T16:20:32.448+02:00  INFO 124278 --- [           main] [
                                          ] o.s.boot.tomcat.TomcatWebSe
rver           : Tomcat started on port 8080 (http) with context path '/'
2025-10-01T16:20:32.449+02:00  INFO 124278 --- [           main] [
                                          ] o.s.s.petclinic.PetClinicAp
plication      : Started PetClinicApplication in 0.595 seconds (process r
unning for 0.765)
```

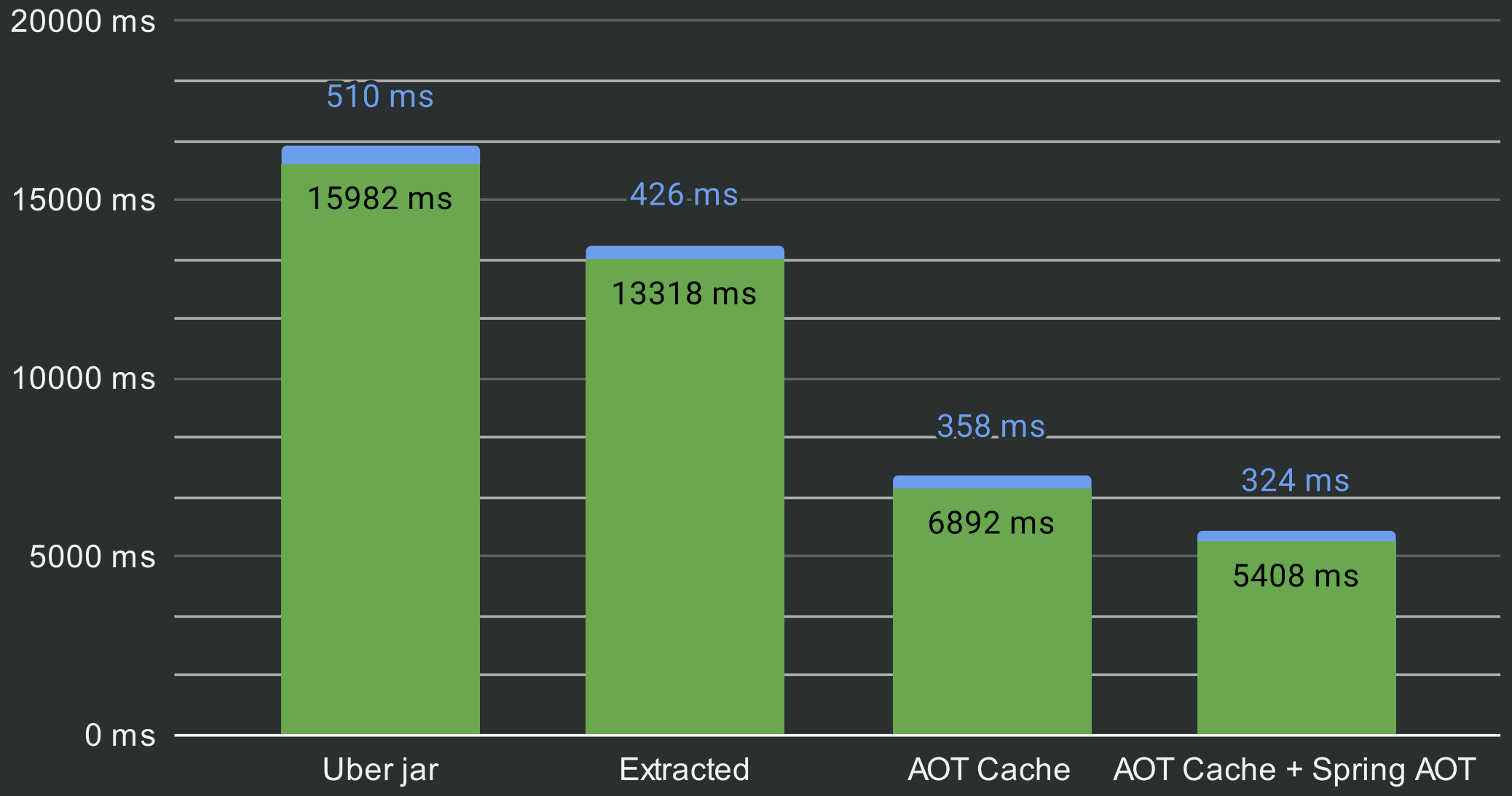
# Time to first request (Spring Boot 4.0.0-M3, Java 25)



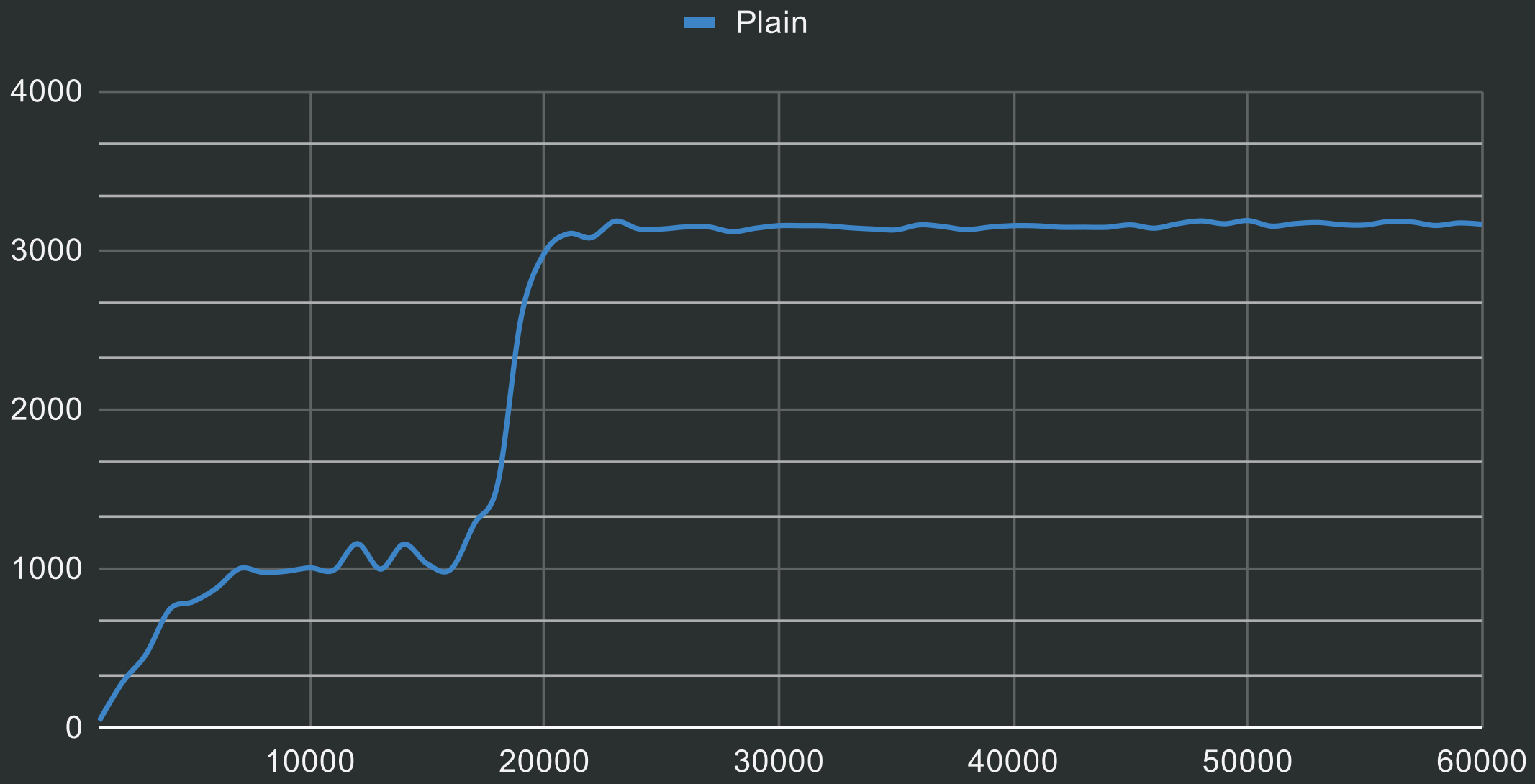
# RSS (Spring Boot 4.0.0-M3, Java 25)



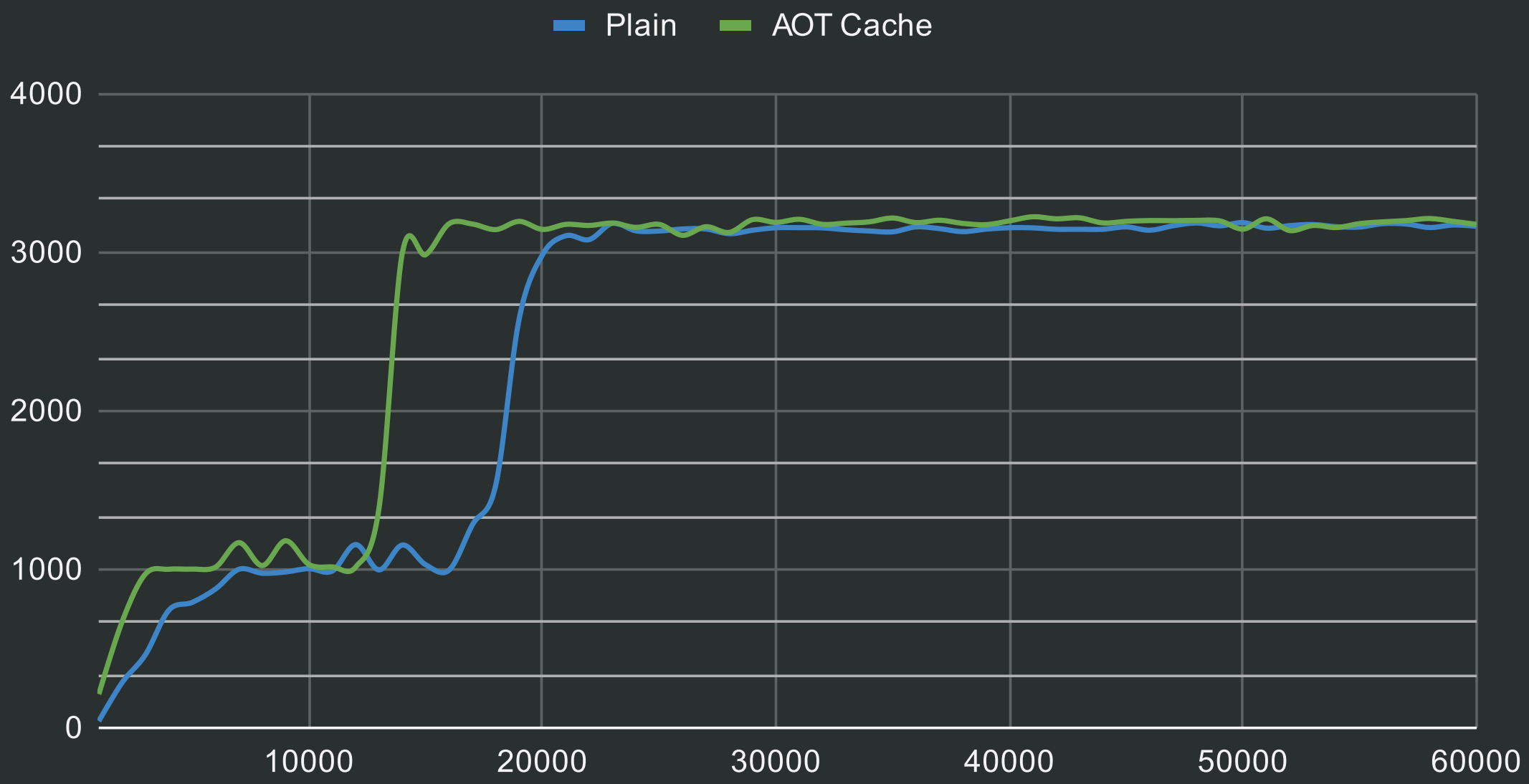
# User time / Kernel time (Spring Boot 4.0.0-M3, Java 25)



# Requests / second (1 CPU)



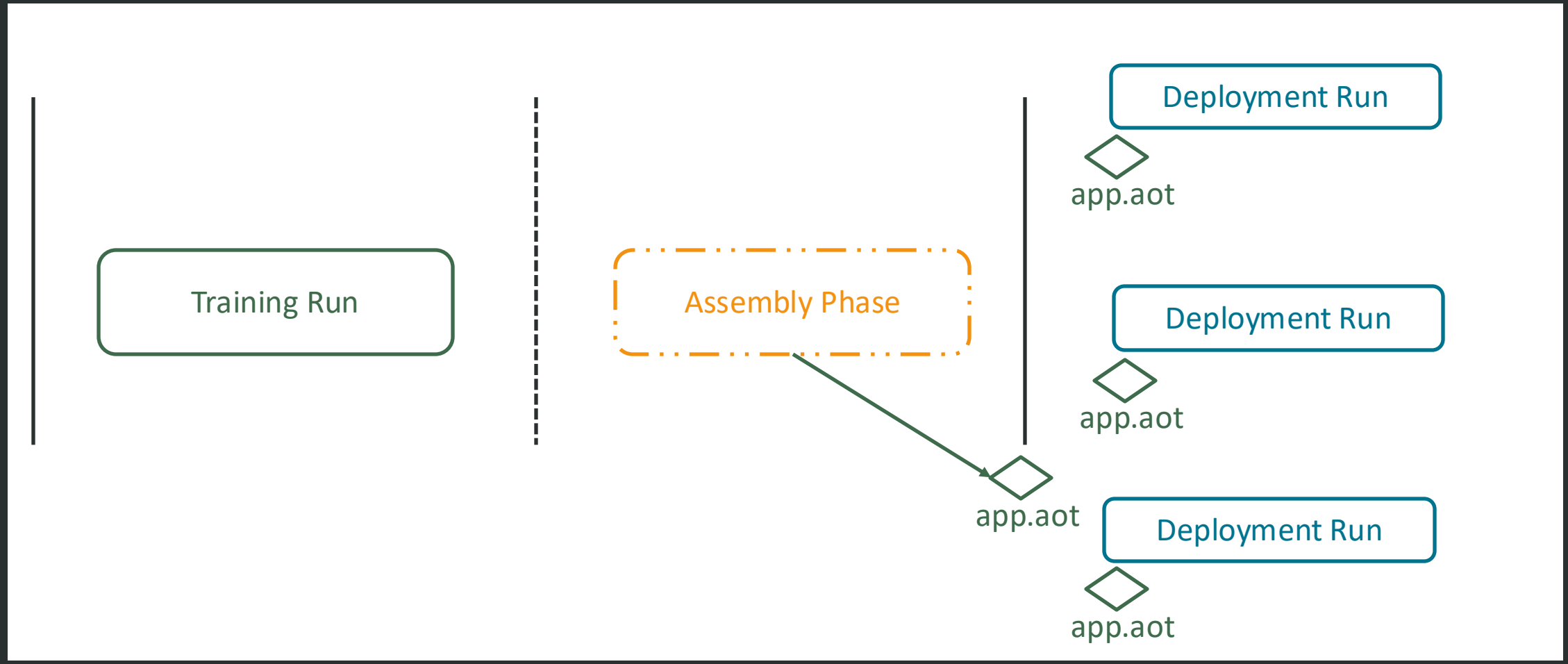
# Requests / second (1 CPU)



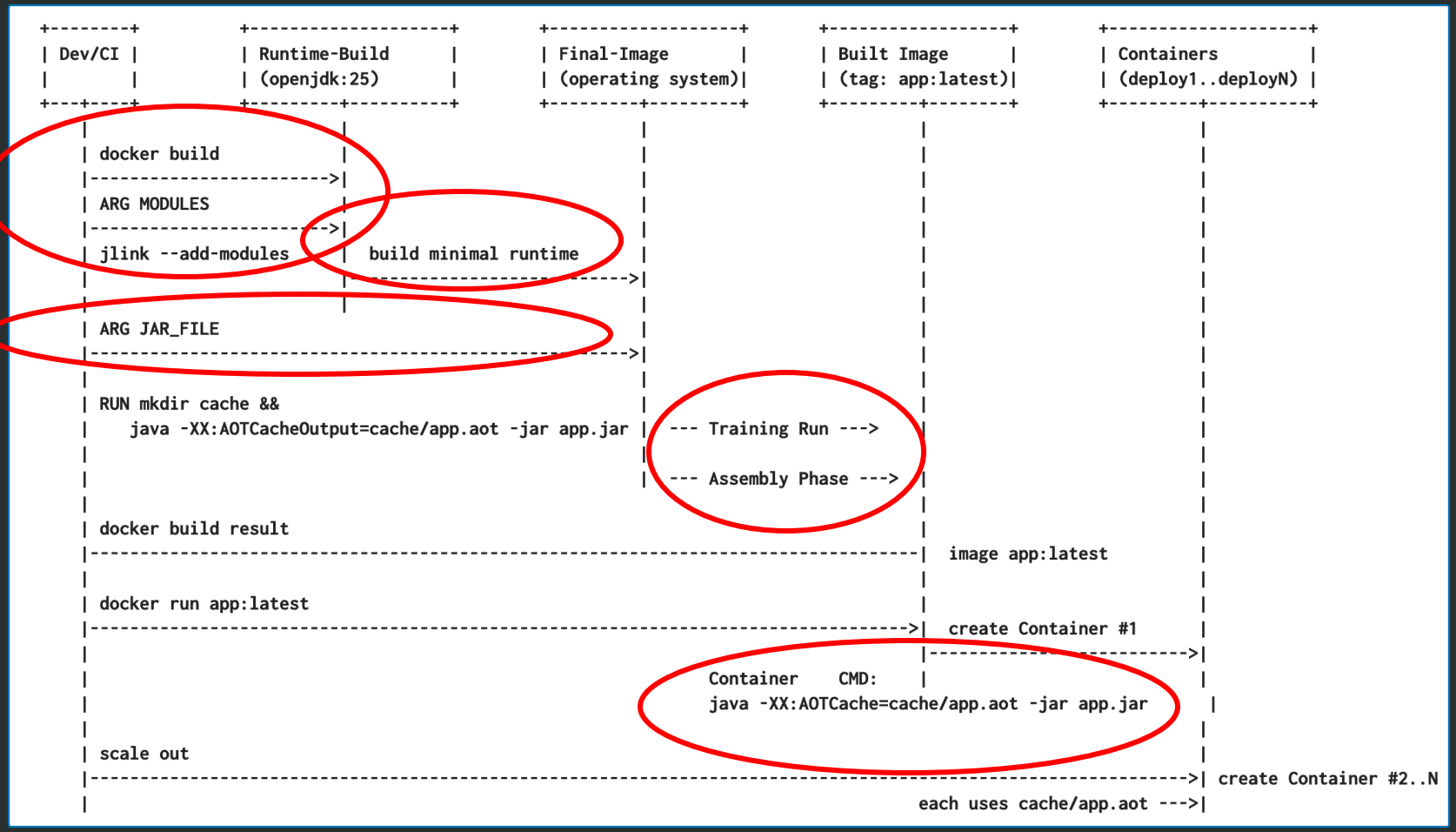

# Operational Approaches

Examples

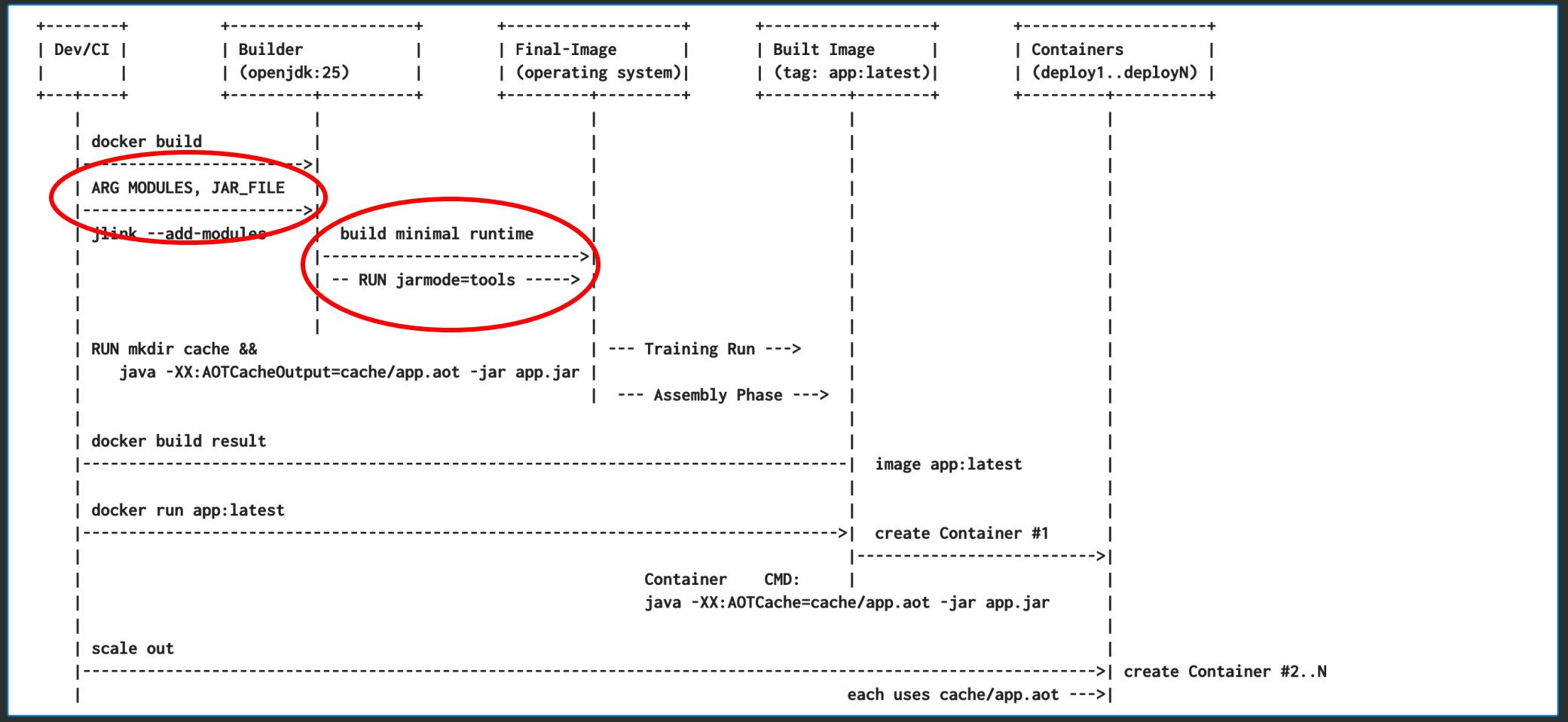
# Training / Assembly / Deployment



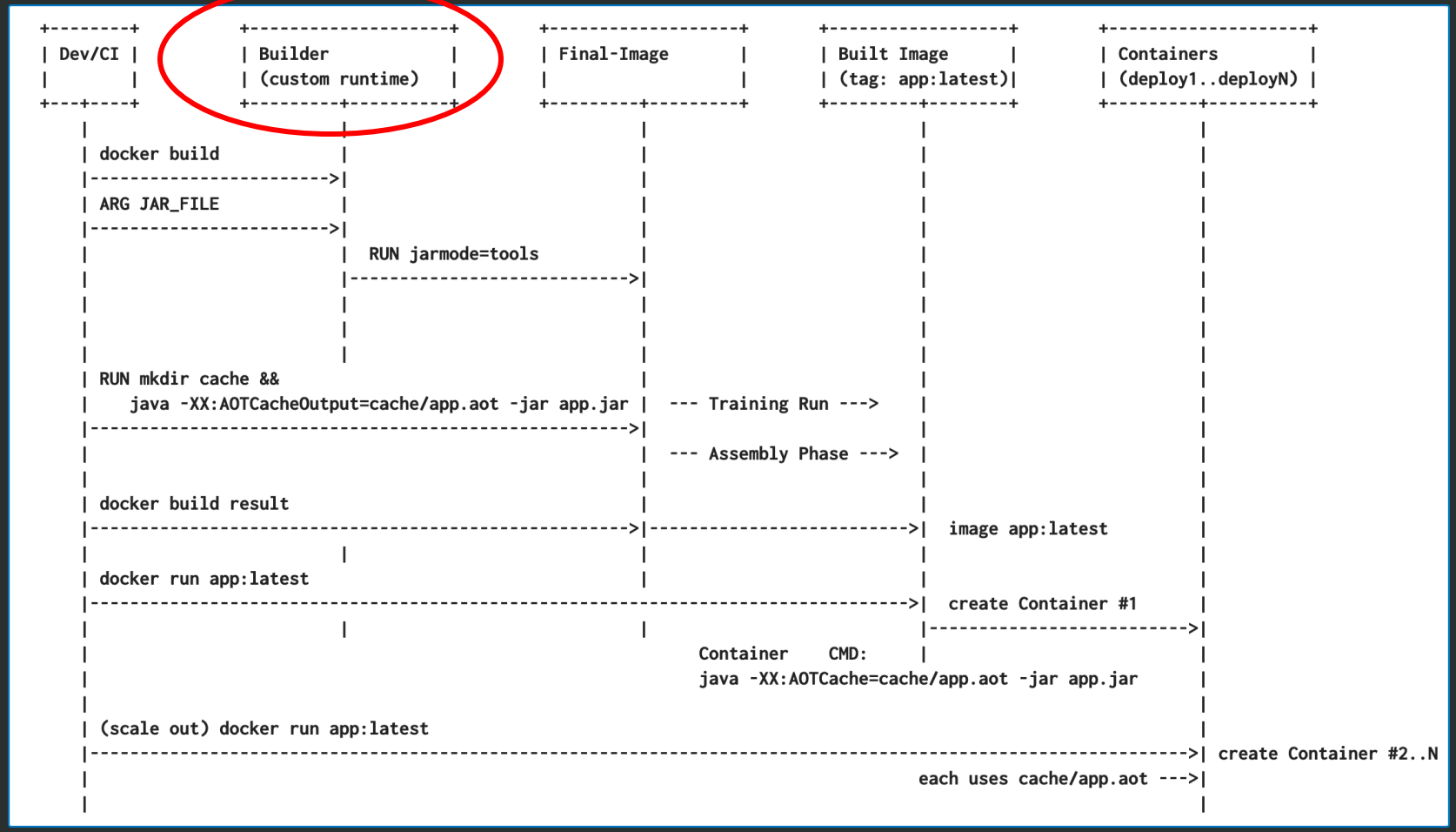
# Two-Step Workflow with JDK Tools (1)



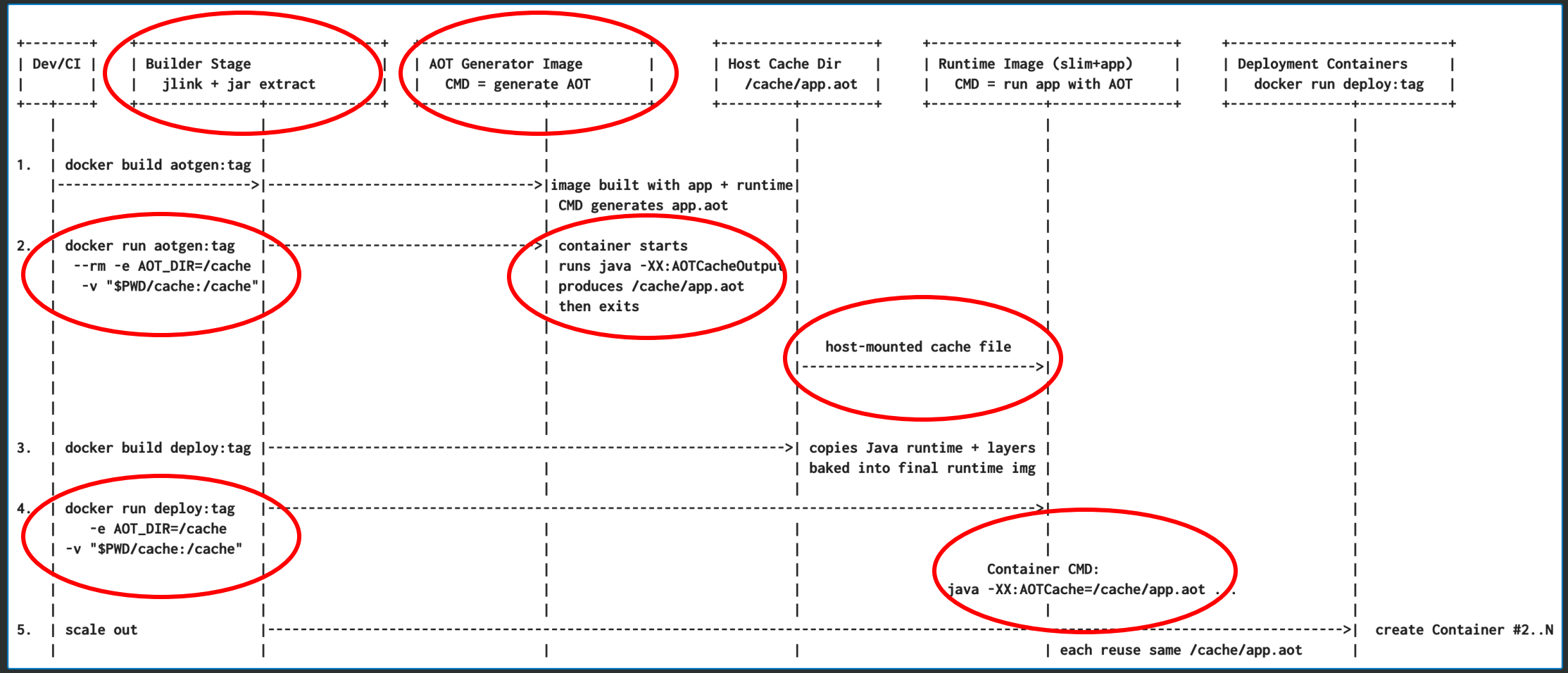
# Two-Step Workflow with JDK Tools (2)



# Two-Step Workflow with JDK Tools (3)



# Two-Step Workflow with JDK Tools (4)




https://gist.github.com/ammbra/59aa7cdb776145a227469730020aa5a4?permalink_comment_id=5780108#gistcomment-5780108

# ⚠ What to Be Careful With?

**Cache Validity / Staleness**

If you rebuild the application or upgrade JDK, you must regenerate the AOT cache.

**Portability**

Make sure you generate AOT with the same base image as runtime.

**Startup Path Coverage**

If your training run is shallow, you won’t warm up enough classes, and cache benefits will be limited.

**Operational Complexity**

Mounting /cache volumes must be correctly handled in Kubernetes/Docker.


# What's Next?

# JEP draft: Ahead-of-Time Code Compilation

| | |
|---|---|
| *Owner* | John Rose |
| *Type* | Feature |
| *Scope* | Implementation |
| *Status* | Draft |
| *Component* | hotspot / compiler |
| *Created* | 2024/06/30 04:47 |
| *Updated* | 2024/08/04 21:52 |
| *Issue* | 8335368 |

## Summary

Enhance the AOT cache to store compiled code from training runs, reducing delays to application startup and warmup.

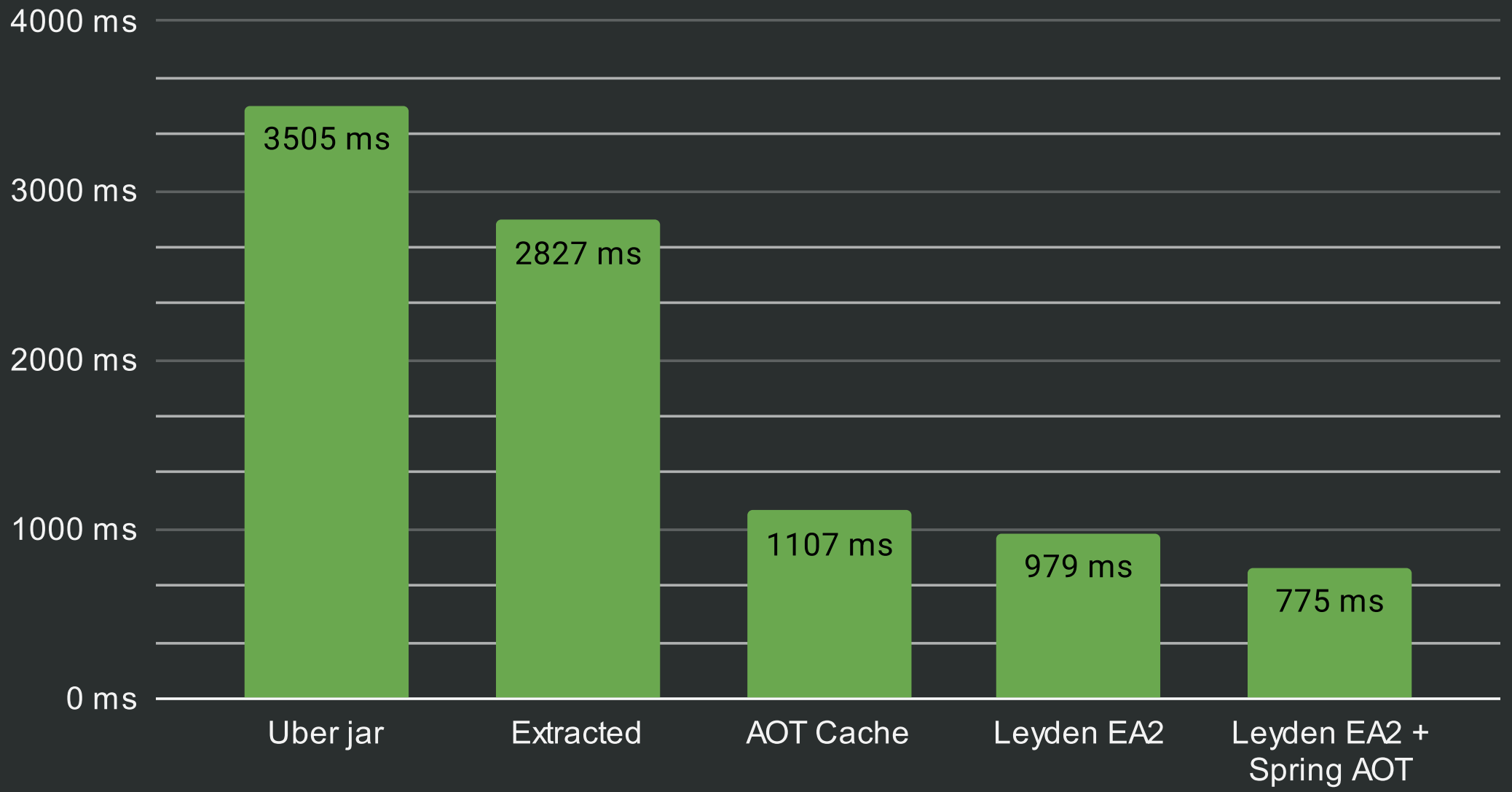
## Goals

- Help applications start up and warm up more quickly, by supplying them with precompiled native code, immediately upon VM startup.
- Preserve compatibility with existing online optimization and compilation technology of the HotSpot VM.

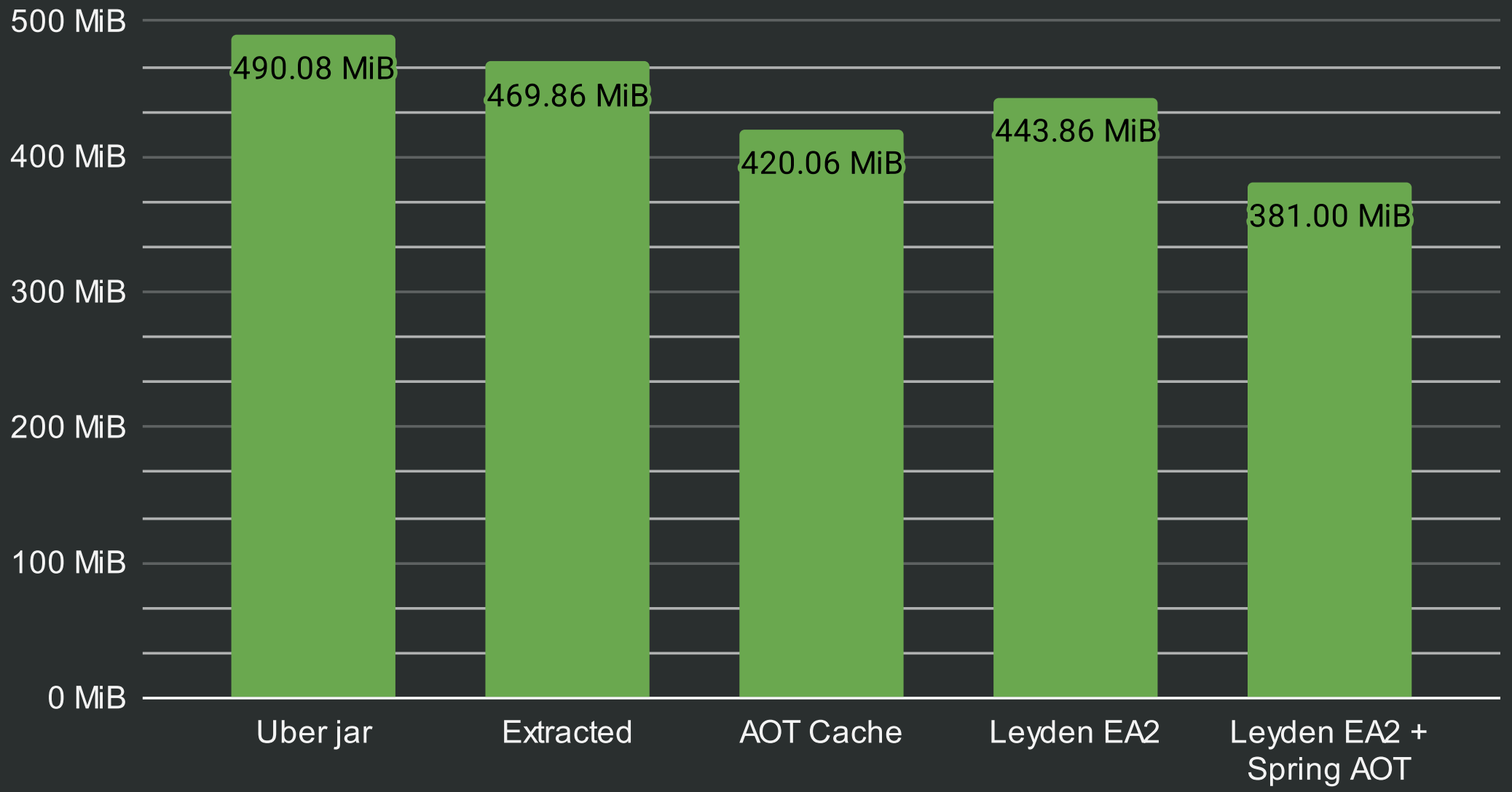
This improvement serves the larger goals of Project Leyden, which include better startup, warmup, and footprint for Java applications.

```
erFactories    : Hibernate is in classpath; If applicable, HQL parser wil
l be used.
2025-10-01T16:15:02.236+02:00  INFO 122541 --- [           main] [
                                           ] o.s.b.a.e.web.EndpointLinks
Resolver       : Exposing 13 endpoints beneath base path '/actuator'
2025-10-01T16:15:02.252+02:00  INFO 122541 --- [           main] [
                                           ] o.s.boot.tomcat.TomcatWebSe
rver           : Tomcat started on port 8080 (http) with context path '/'
2025-10-01T16:15:02.254+02:00  INFO 122541 --- [           main] [
                                           ] o.s.s.petclinic.PetClinicAp
plication      : Started PetClinicApplication in 0.454 seconds (process r
unning for 0.621)
```

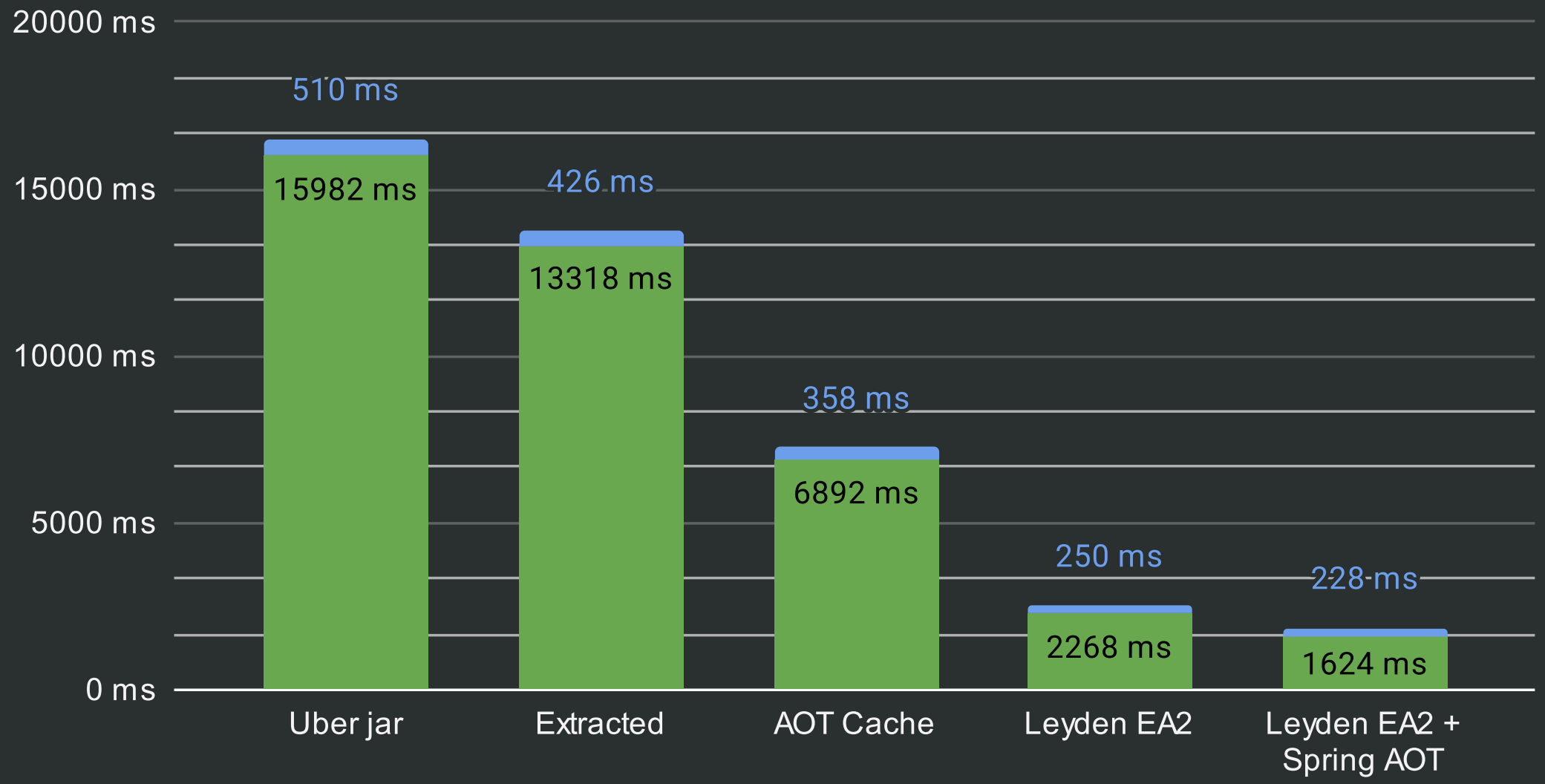
# Time to first request (Spring Boot 4.0.0-M3, Java Leyden EA2)



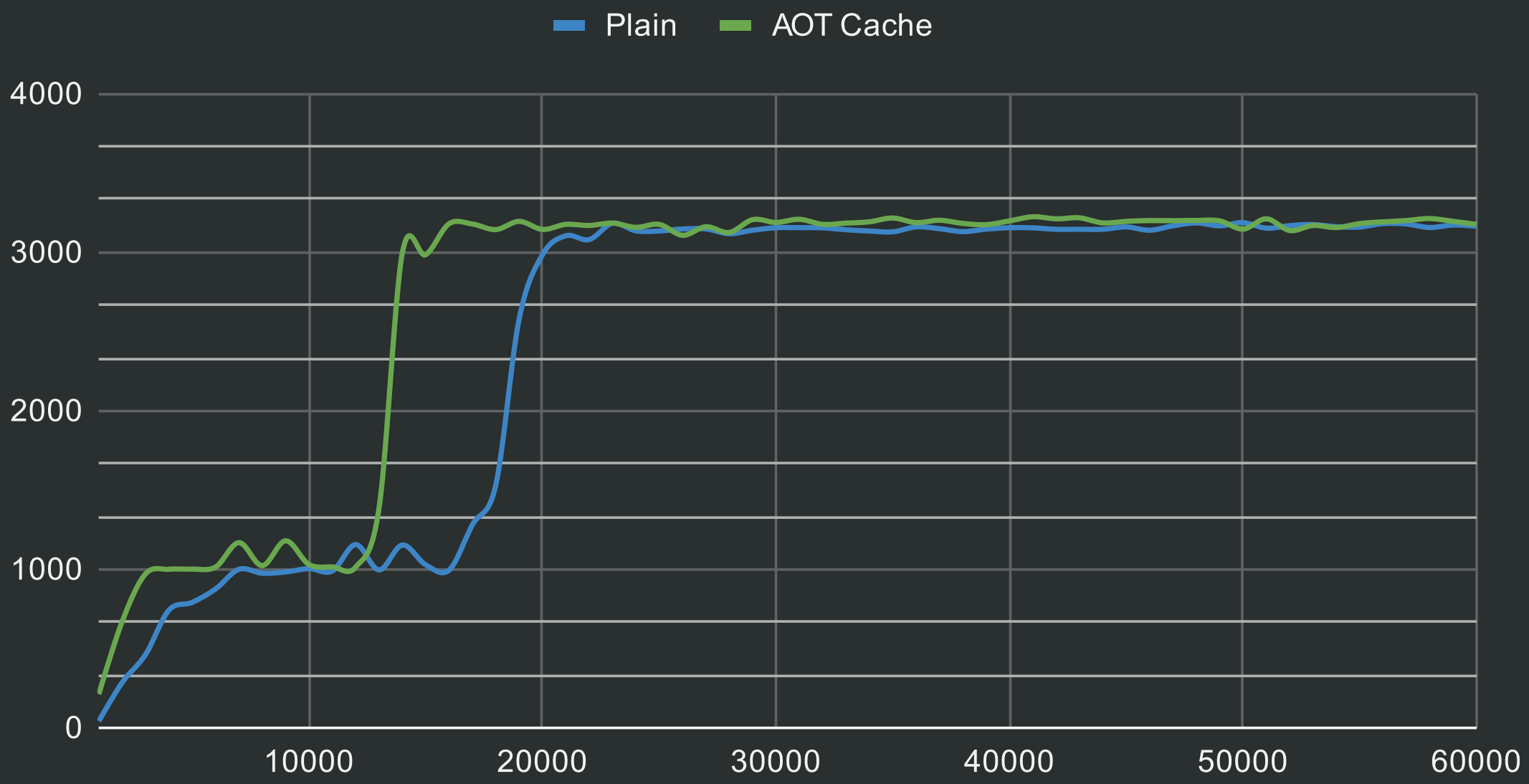
# RSS (Spring Boot 4.0.0-M3, Java Leyden EA2)



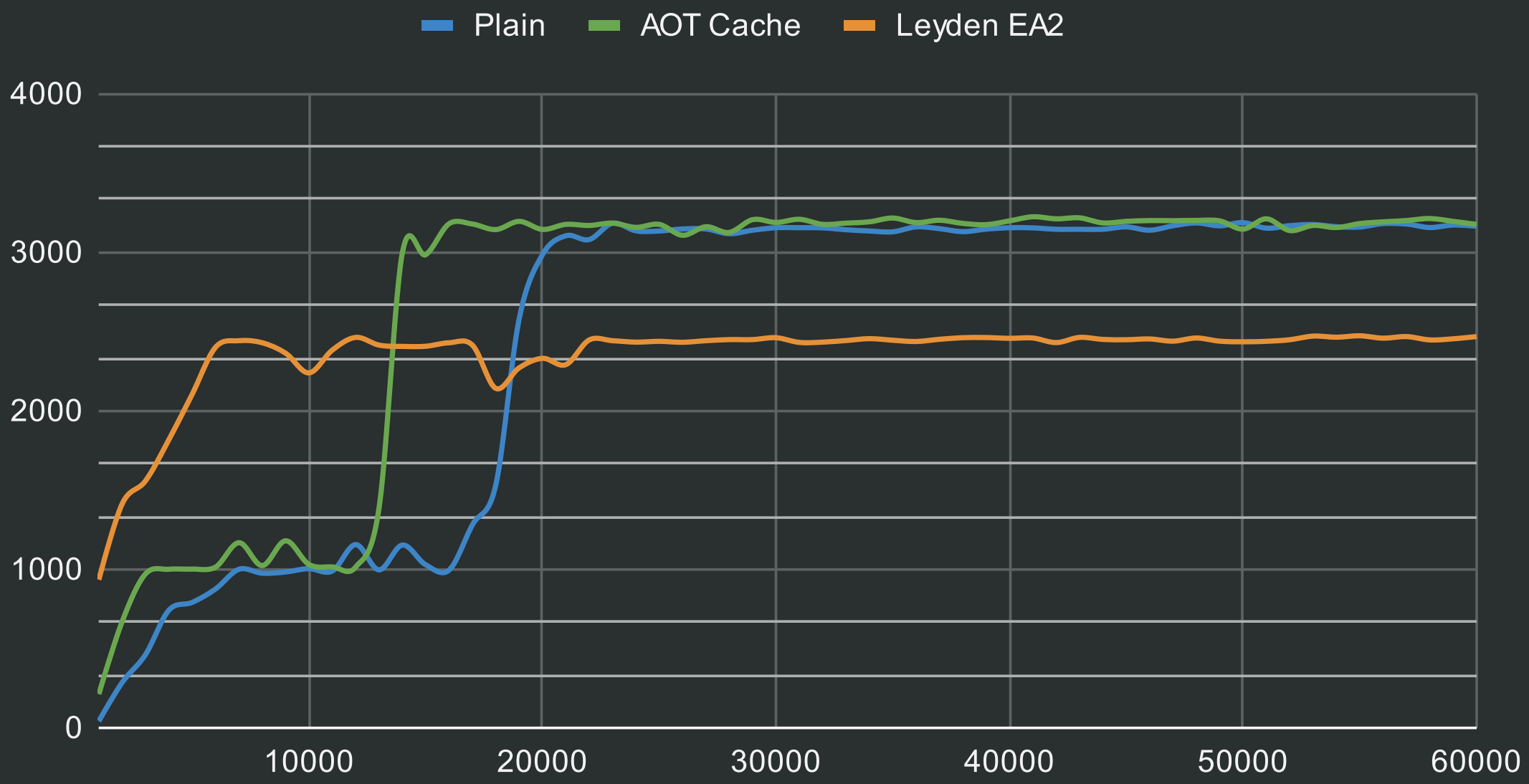
# User time / Kernel time (Spring Boot 4.0.0-M3, Java Leyden EA2)



# Requests / second (1 CPU)



# Requests / second (1 CPU)



# Balance of Performance and Portability in Leyden

**Portability is a priority.**

- Should AOT code use the best possible instructions or compile for a “typical” deployment target?
- Also applies to other parts of the system as well (heap sizes, garbage collectors, etc)

**Extend AOT capabilities to user-defined classloaders.**

- Training runs are currently focused on the 3 built-in classloaders (System, Extension, and Boot loaders) and the classes loaded by them.
- User classloaders would benefit from the same concept but there a lot of challenges to get there.




# AOT improvements planned in Spring projects

- Introduce modular Spring AOT
  - Pick and choose which parts of Spring AOT are used
  - e.g. you may want Spring Data repositories and predefined classes, but you don't want a frozen bean arrangement#
- Precompute classpath scanning
- Customized condition evaluation
- Add hook points for bean optimization
- …

Current prototype:

```
processAot {
    beanRegistration = false
    predefinedClasses = true
    classpathIndexes = true
    reachabilityMetadata = false
}
```

**Invest in training your application today.**

**Keep up with the JDK and Spring Boot releases to unlock available optimizations.**

**Get better performance each release, with Leyden fully Java Platform–compatible.**


# Thank you





# Sign up for updates! javaone.com