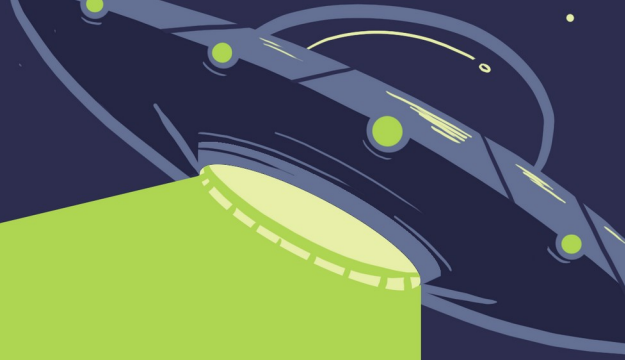
Open liberty



# Instantly startup your Spring and Jakarta applications ... what a CRaCing idea!

Thomas Watson

JavaMUG – February 14th, 2024

# Who am I

- Senior Software Engineer at IBM
- Based in Austin, TX USA
- Focus on OSS cloud native Java technologies
- Long history in Eclipse and Apache foundations

Thomas Watson
@TomWatson5150
tjwatson@us.ibm.com

# Cloud Challenges

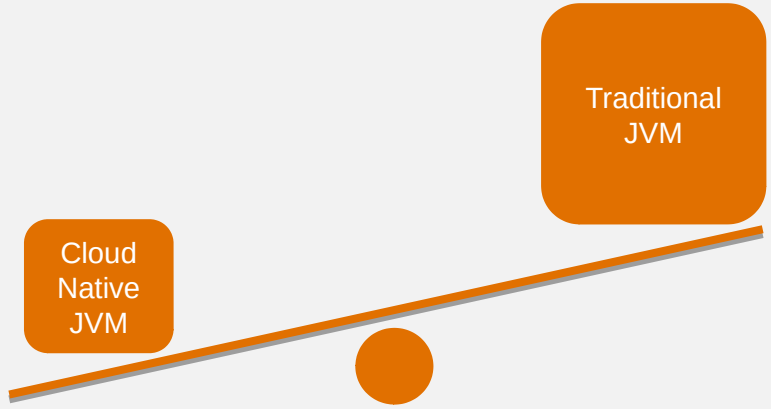
# Challenges of the Cloud-native era

The shift to cloud-native has changed the demands placed on application frameworks and the underlying JVM technologies


- Developer Delight
  - Help developers be productive developing cloud native applications
- Cloud Agility
  - Enable business innovation through accelerated cloud-native application delivery
- Operational Efficiency
  - Help businesses reduce costs and achieve sustainability goals

# Cloud Economics

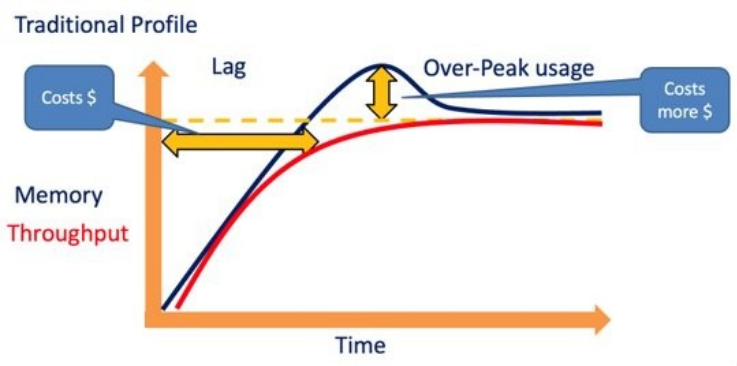
- Pay-as-you go pricing model
  - Based on CPU and Memory usage
- Elastic scaling based on demand is crucial to keep costs low
- Requirements
  - Scale-to-zero
  - Low memory usage
  - Minimal latency – fast startup

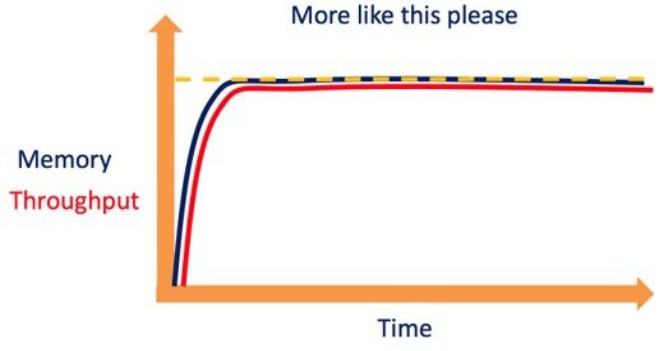


# Java Startup



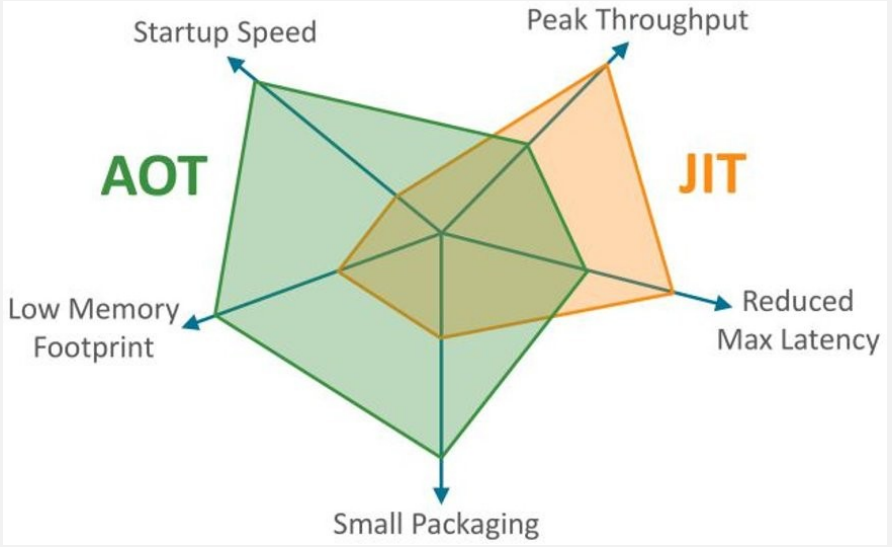
- Additional resources needed to get to peak performance
  - JIT requires additional memory and CPU to warm up
- Once warmed up the runtime can provide very good performance for both memory and throughput
  - Garbage collector does its job
  - JIT-compiled code optimizes high traffic code
- Can we skip the ramp-up required for Java?
  - Achieve immediate peak performance



# AOT vs JIT Compilation

- Ahead of Time, static compilation
  - Closed world assumption allows small packaging
  - No JIT runtime overhead – less memery needed
  - Extremely fast startup
  - Compromises on full Java functionality

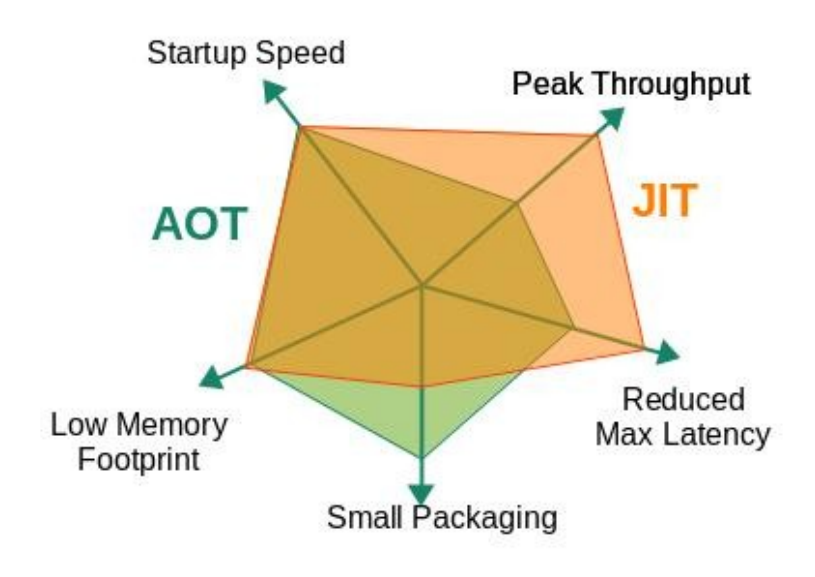



https://twitter.com/thomaswue/status/1145603781108928513?s=20&t=-6ufSBjc46mfN5d_6Y2-Rg

# Can we get here?

Open Liberty

- Achieve Rapid Startup
- Avoid JIT cost during ramp-up
- Keep all the benefits of full JVM
  - Peak Throughput
  - Highly optimized garbage collector
  - Keep all other features of Java



# Checkpoint / Restore

# Liberty InstantOn


Liberty InstantOn is a holistic solution that provides fast startup without compromise

- Use Checkpoint / Restore
  - CRIU Project – https://criu.org
  - Linux only
- Select point in startup to checkpoint
  - Restore application from checkpoint

- Semeru Runtimes
- Checkpoint / Restore with Semeru InstantOn
- Where to checkpoint?
- Container support

# Where to Checkpoint?

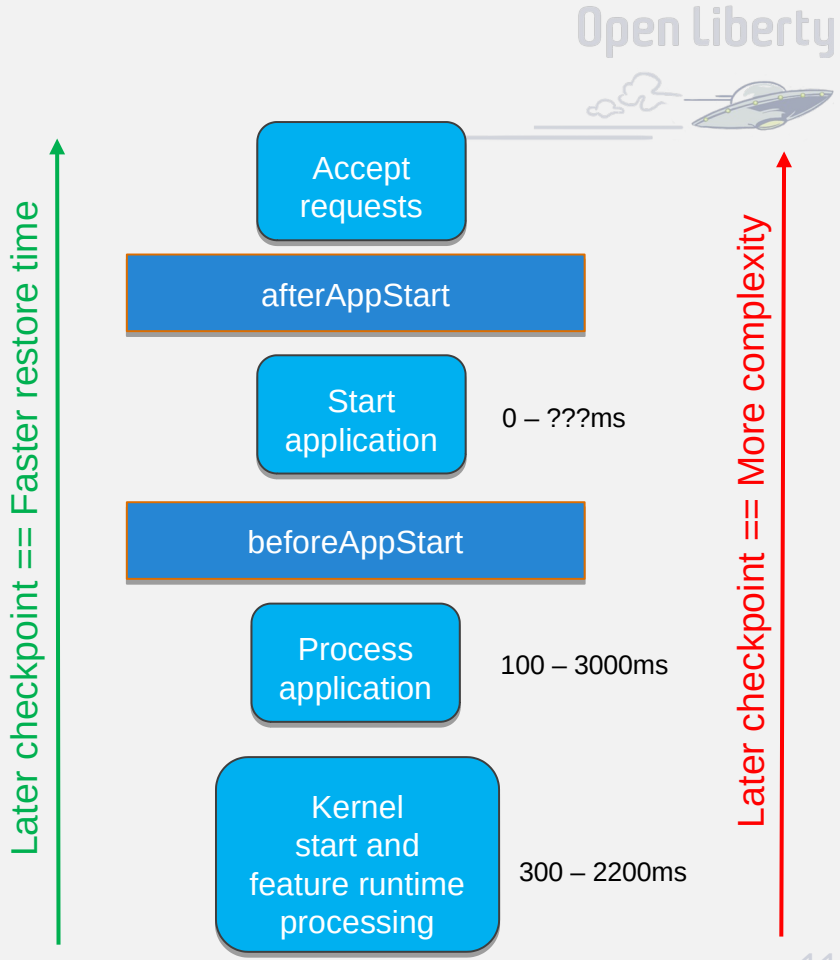
Tradeoff between startup time and restore complexity

Liberty InstantOn offers two options

- beforeAppStart
- afterAppStart

Choice depends on application behavior during startup

- Access remote resources
- Read configuration that is expected to change at deployment
- Start a transaction

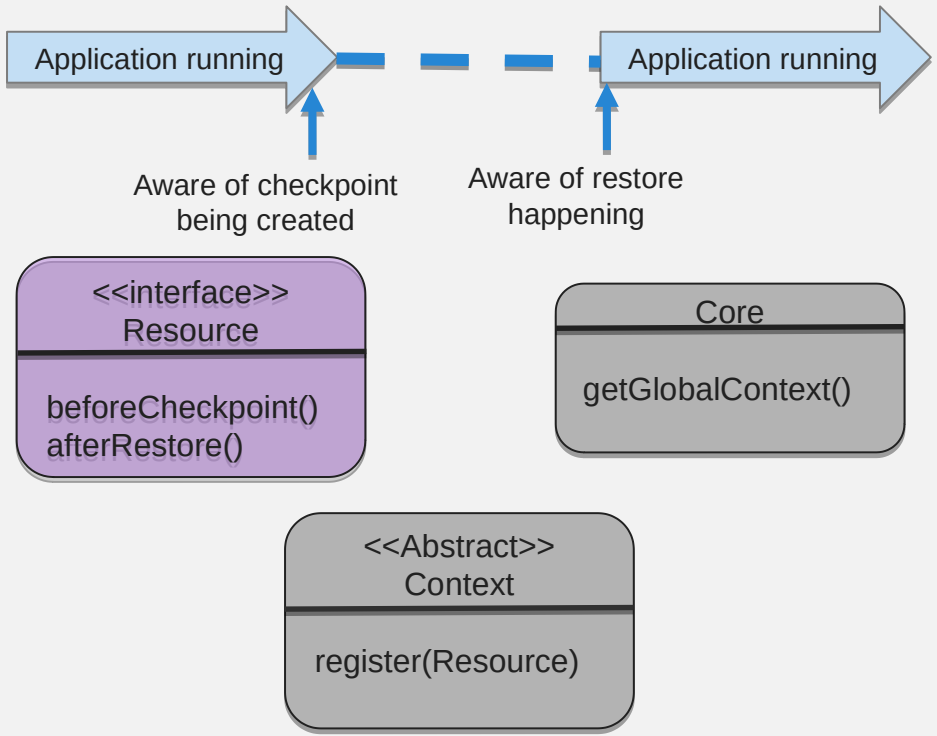


# Demo

https://github.com/tjwatson/guide-getting-started/tree/InstantOn

# CRaC – Coordinated Restore at Checkpoint

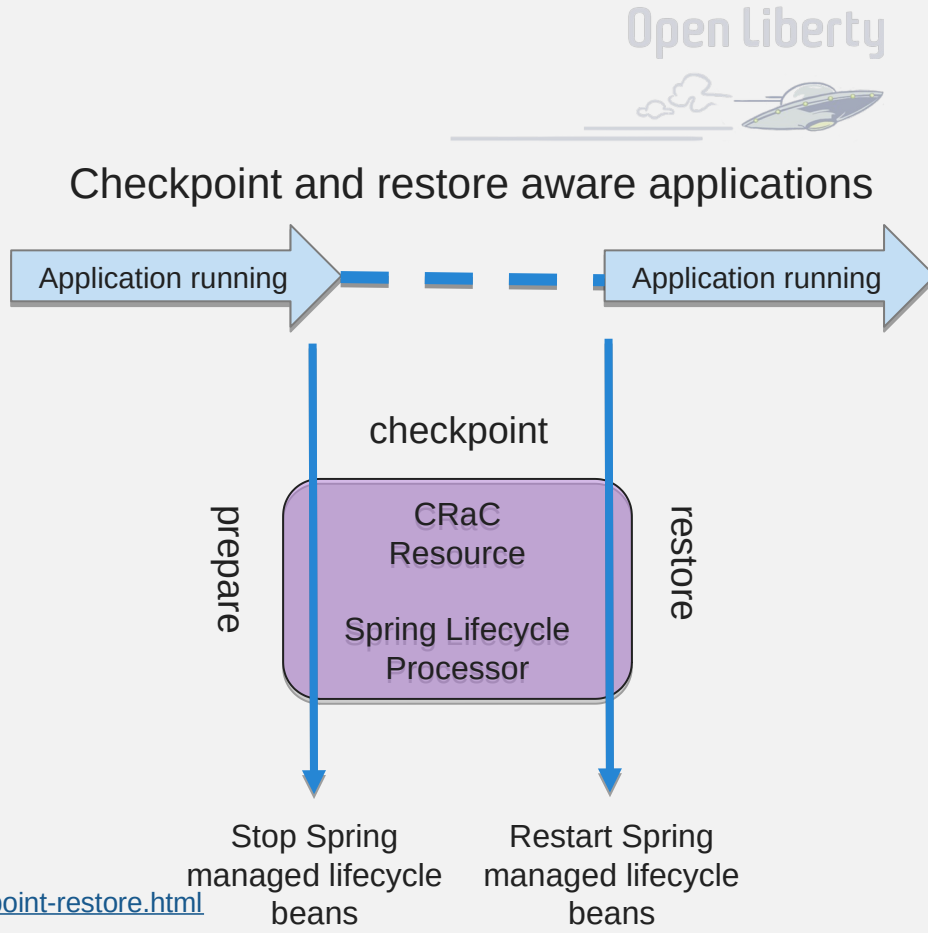
- Project org.crac - Provides APIs to smoothly transition to a CRaC solution [1]
- Allows applications to participate in the checkpoint / restore process
- Compile against and use at runtime, even without a CRaC enabled JVM




[1] https://github.com/CRaC/org.crac

# Spring - Coordinated Restore at Checkpoint

- Recent Spring Framework 6.1release integrates with checkpoint / restore using org.crac [1]
- Spring Lifecycle contract hooks into checkpoint / restore using org.crac resources
- Open Liberty 23.0.0.10-beta offers org.crac, enabling Spring Boot applications to use InstantOn [2]




[1] https://docs.spring.io/spring-framework/reference/6.1/integration/checkpoint-restore.html
[2] https://openliberty.io/blog/2023/09/26/spring-boot-3-instant-on.html

# Demo

https://github.com/tjwatson/spring-petclinic/tree/LibertyInstantOn

# The End

Open Liberty