From CLIs to LLMs: 5 Cool Applications to Build with GraalVM

Alina Yurenko

Oracle

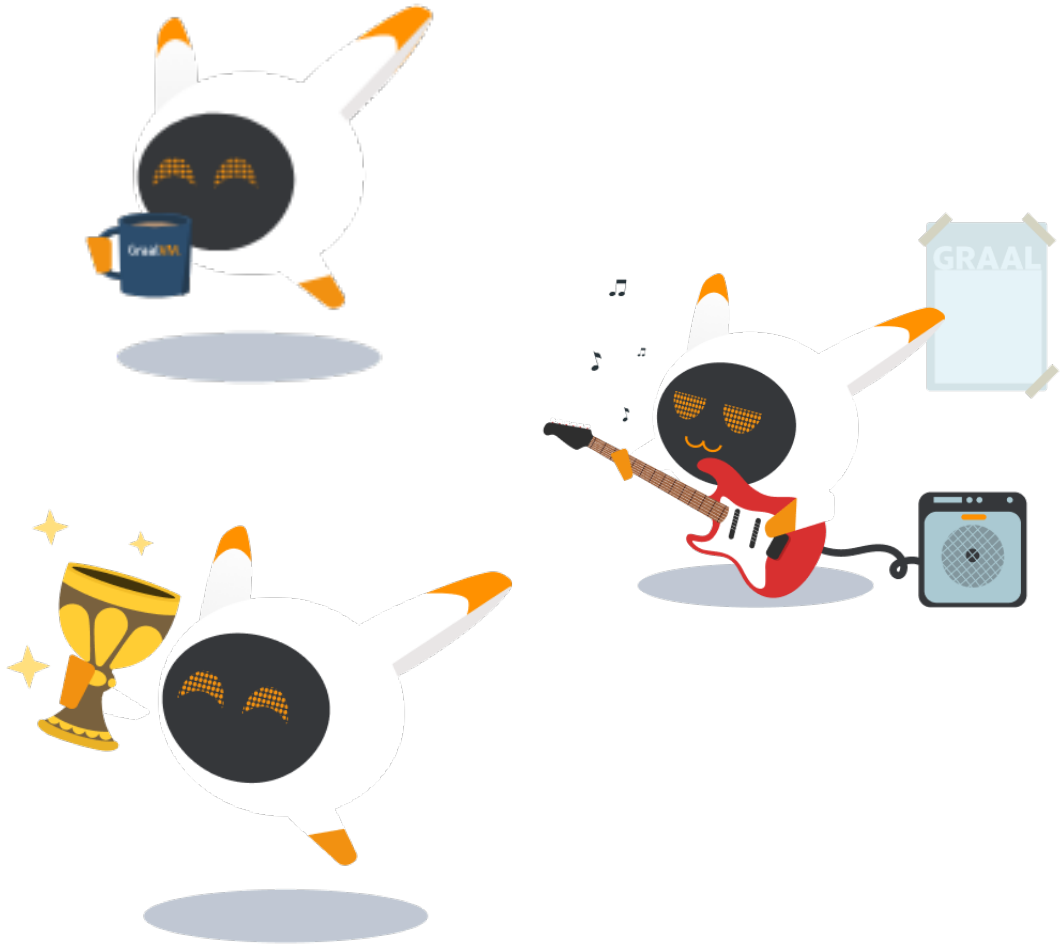
GraalVM™

About me

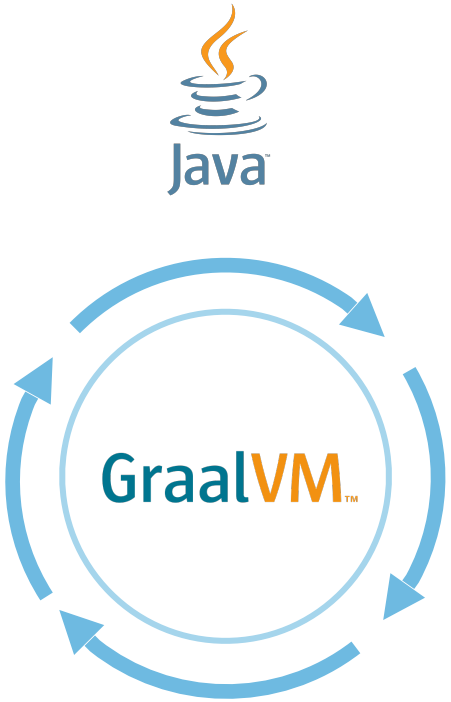
- Alina Yurenko / @alina_yurenko
- Developer Advocate for GraalVM at Oracle
- Love open source and communities 🤝
- Love both programming 👩‍💻 & natural languages 🗣️

The Agenda 🐰

- Intro
- CLI application 🤖
- Full-Stack web app 🌎
- An LLM inference engine 🤯
- AOT at the speed of JIT 🚀
- An extra secure application 🛡️
- Bonus application 🐍
- Future plans 🚀
- Get started & links



What you can do with GraalVM!



High performance with Graal JIT

AOT with Native Image

Extend Java with other languages

Why GraalVM Native Image?

Start
& Scale Fast

Low Resource
Usage

Predictable High
Performance

Improved
Security

Compact
Packaging

Ecosystem
Support

1. A CLI Application

Start
& Scale Fast

Low Resource
Usage

Compact
Packaging

How?

- AOT-compiled native executables are perfect for distributing CLI apps: small, fast, minimum overhead
- Notable mention:  picocli

  Sample project: github.com/alina-yur/native-spring-shell

2. A Full-Stack Application

Start
& Scale Fast

Low Resource
Usage

Compact
Packaging

Predictable High
Performance

How?

- Compile and bundle your application as a native image for fast startup and easy distribution
- Notable mentions:  Vaadin,  JHipster
- Sample projects:

  github.com/alina-yur/native-spring-ai

  github.com/marcushellberg/travel-tips

3. A Complete LLM Inference Engine

Start
& Scale Fast

Low Resource
Usage

Compact
Packaging

Pure Modern
Java

How?

- Fast LLM inference in pure modern Java
- No dependencies (73kB jar file)
- FFM API and Vector API for quick native access and quick computations
- GraalVM makes it even faster 🚀

  Native Image support with AOT model pre-loading for instant time-to-first-token

  The project: github.com/mukel/llama3.java

4. AOT at the speed of JIT 🚀

Predictable High Performance

Low Resource Usage

How?

- **Profile-guided optimizations**

Collect and use profiles to optimize for the specific runtime behaviour of your application

- **ML-enabled PGO**

Use a ML model to automatically predict the profile of the application

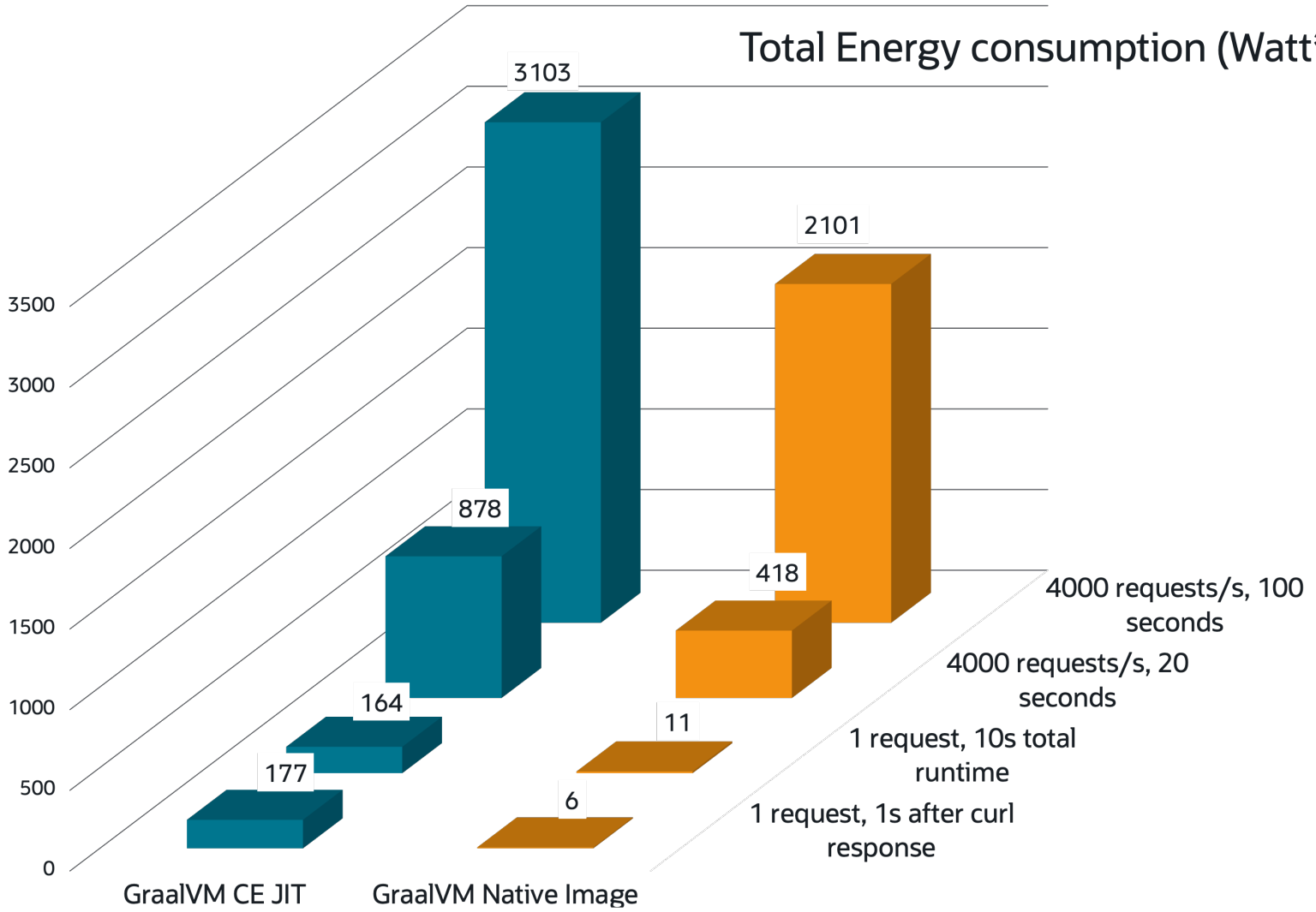
- **G1 GC**

Optimize GC for peak throughput and improved latency

- **`-march=native`**

Optimize for the specific hardware features of the machine you'll be running on

4A. Spring PetClinic — Energy Consumption



5. Extra Secure Application

Improved Security

How?

- Reduced attack surface area due to dead code removal—unused classes, methods, and fields not included in executable
- SBOM supporting industry standards
  Embedded in executables
  CycloneDX format
- Not vulnerable to deserialization attacks
  via class loading—executable includes only required and specified classes
- Not vulnerable to JIT compiler attacks
  all code is AOT compiled

GraalVM for JDK 24 🚀

- Java 24 features 😍
- The fastest GraalVM yet :)
- New ML profile inference for even higher performance out of the box 🤖
- Smaller executables 📦
- Vector API support in Native Image 🚀
- New SBOM features 🛡️
- Developer experience improvements 👩‍💻
  Learn more: medium.com/graalvm

Get started with GraalVM 🚀

graalvm.org

docker pull container-registry.oracle.com/graalvm/jdk:24

sdk install java 24.0.1-graal

github.com/graalvm/graalvm-demos

GraalVM™

Thank you!

Alina Yurenko
@alina_yurenko