Green Coding with Spring Boot

Sustainability as a Path to Better Software

Patrick Baumgartner
42talents GmbH, Zürich, Switzerland

@patbaumgartner
patrick.baumgartner@42talents.com

Abstract

Green Coding with Spring Boot: Sustainability as a Path to Better Software

Green coding is more than just a trend – it is a **promise of quality**.

Those who focus on **resource-efficient software** naturally make better choices: more efficient code, more stable systems, and lower operating costs. This session shows how to achieve that with **Spring Boot**.

We focus on **architectural principles** and **performance optimisations** that save energy – while also making the codebase **leaner, more maintainable, and more scalable**. This includes choosing the right **data structures and algorithms**, purposeful use of **Spring Boot components**, and optimising **servers and cloud infrastructures**.

For everyone who wants to develop not only in a *green* way – but above all in a **better** way.

Abstract

Green Coding mit Spring Boot: Nachhaltigkeit als Weg zu besserer Software

Green Coding ist mehr als ein Trend – es ist ein **Qualitätsversprechen**. Wer auf **ressourcenschonende Software** achtet, trifft automatisch bessere Entscheidungen: effizienterer Code, stabilere Systeme, geringere Betriebskosten. Diese Session zeigt, wie du mit **Spring Boot** genau das erreichst.

Im Fokus stehen **Architekturprinzipien** und **Performance-Optimierungen**, die Energie sparen – und zugleich die Codebasis **schlanker, wartbarer und skalierbarer** machen. Dazu gehören die Auswahl geeigneter **Datenstrukturen und Algorithmen**, der gezielte Einsatz von *Spring-Boot-Komponenten** sowie die Optimierung von **Servern und Cloud-Infrastrukturen**.

Für alle, die nicht nur (aber auch) *grün* entwickeln wollen – sondern vor allem **besser**.

Green Coding with Spring Boot

Sustainability as a Path to Better Software

Patrick Baumgartner - 42talents GmbH, Zürich, Switzerland

Green Coding mit Spring Boot

Nachhaltigkeit als Weg zu besserer Software

Patrick Baumgartner - 42talents GmbH, Zürich, Schweiz

Who am I?

Patrick Baumgartner

Technical Agile Coach @ **42talents**

I focus on building **software solutions *with* humans**.

Coaching, **Architecture**, **Development**, **Reviews**, and **Training**

Lecturer @ **Zurich University of Applied Sciences (ZHAW)**

Co-Organiser of Voxxed Days Zurich, **JUG Switzerland**, **Java Champion**, Oracle ACE Pro Java ...

@patbaumgartner

Agenda

Agenda

- Why **green coding** matters
- Principles of **green software**
- **Measure first**: performance & energy
- **Architecture** choices in Spring Boot
- **Code-level practices** (DB, APIs, JVM, Loom, Native Image)
- **Infrastructure & scaling** choices
- **People & culture**
- **Takeaways**

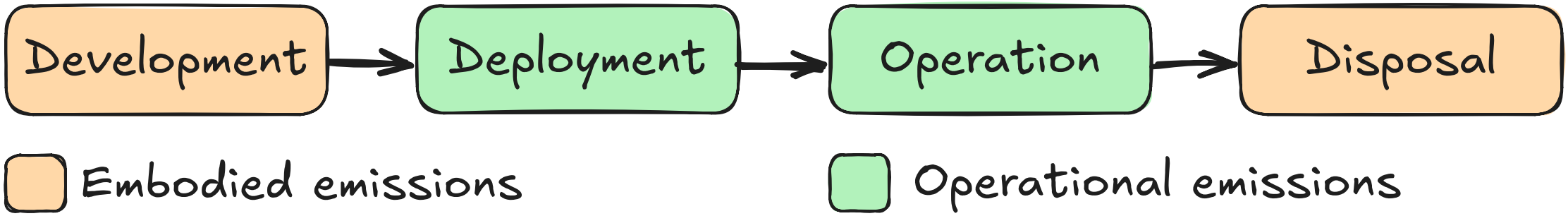
Motivation

ICT’s Carbon Footprint

- ICT responsible for ~**3–4%** of global emissions
- Could reach **10–14% by 2040** if unchecked
- Breakdown:
  - End-user devices: **32–57%**
  - Networks: **22–25%**
  - Data centres: **18–41%**
- ~**25%** of footprint = **embodied emissions** (hardware production)
- Impact comparable to **aviation** or **agriculture**

Where Do Software Emissions Come From?

- **Embodied**: hardware production, dev laptops, CI/CD infra
- **Operational**: data centres, networks, end-user devices
- Greener code cuts **operational emissions** and extends hardware lifetimes → lowers **embodied emissions**



The Opportunity in Green Coding

- Inefficient code = **wasted CPU cycles**, higher bills, more emissions
- Green coding improves:
  - **Performance** → faster responses, better throughput
  - **Stability** → leaner, less complex systems
  - **Cost-efficiency** → lower compute & storage needs

> Treat **energy efficiency** as a **software quality attribute**

Principles & Architecture

Core Principles of Green Software Architecture

- **Design lean systems** – avoid unnecessary layers
- **Efficient algorithms & data structures** → less CPU & memory
- **Right-size workloads** – no over-provisioning
- **Measure energy continuously**
- **Lifecycle view** – optimise runtime, maintainability, scalability

Every CPU cycle
and every byte transmitted
has a carbon cost ...

Architectural Trade-offs

- **Monoliths**
  - Less orchestration, fewer network hops
  - Simpler deployment
- **Microservices**
  - Independent scaling, flexibility
  - Overhead from inter-service traffic, infra complexity

Rule of thumb: if not needed → keep it **modular monolith**

Optimisations in Practice

Smarter Threads with Project Loom

- **Today**: OS threads expensive, block on I/O
- **Loom**: lightweight **virtual threads** (Java >21)
- Benefits:
  - Better CPU utilisation
  - Lower memory footprint
  - High concurrency without reactive complexity

Watch-outs: avoid pinning (`synchronized`, blocking drivers)

Native Images & AOT with Spring Boot 3.x

- **GraalVM native image** → very fast startup, low memory
- Best for **serverless**, **scale-to-zero**, or **spiky loads**
- Trade-offs:
  - Longer builds, reflection config
  - Sometimes lower peak throughput

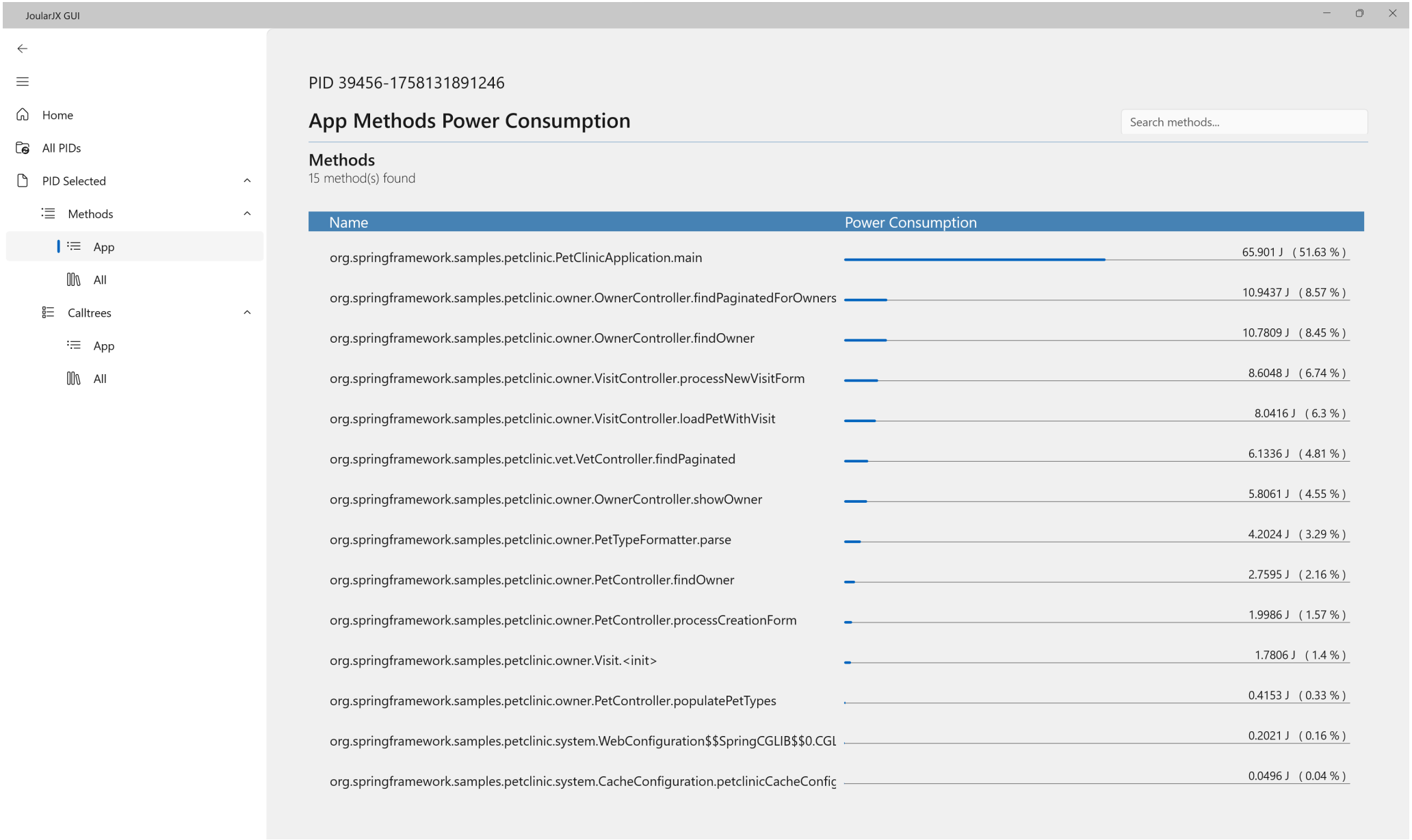
Rule: use **JVM for throughput**, **native for cold-starts**

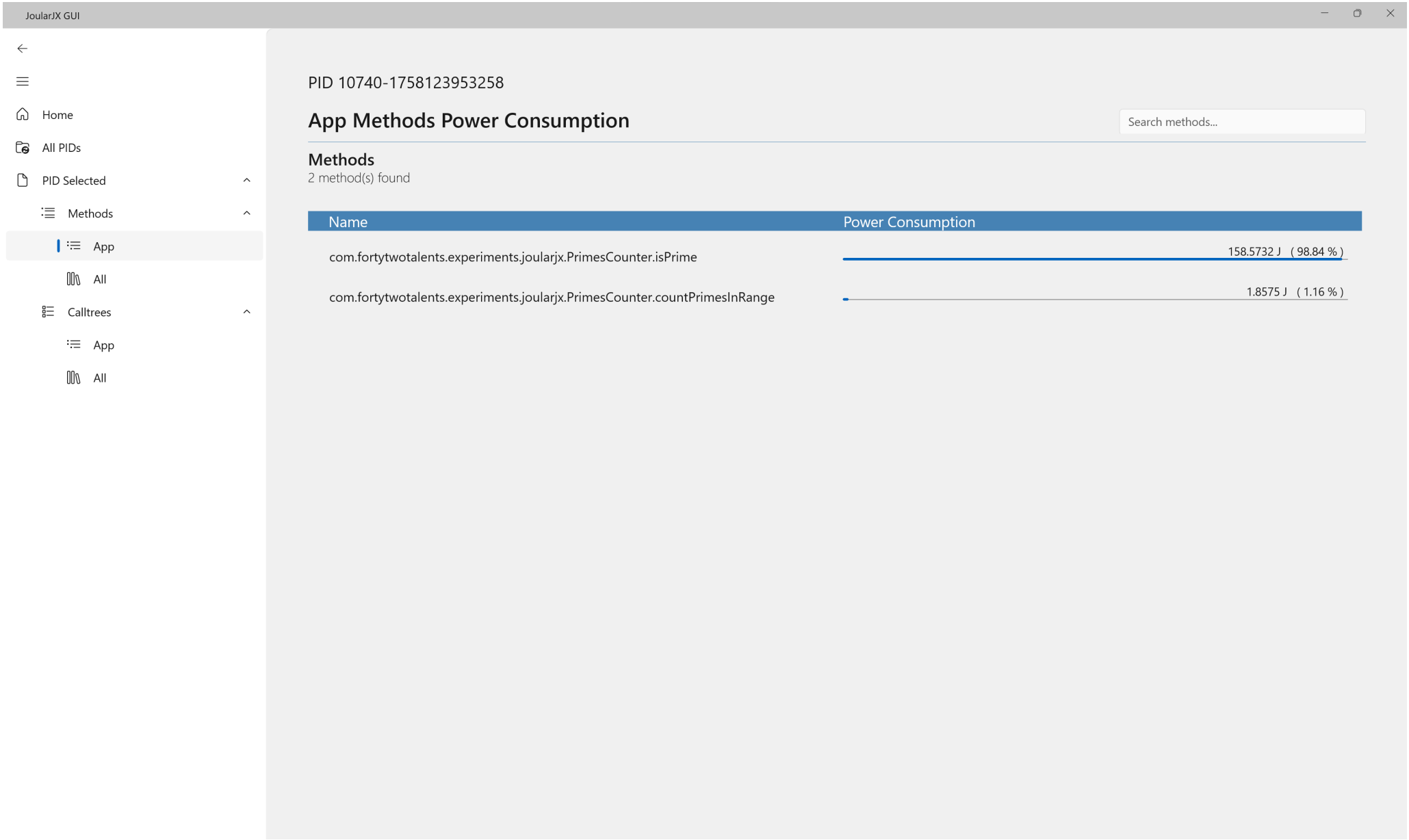
Performance & Energy Metrics

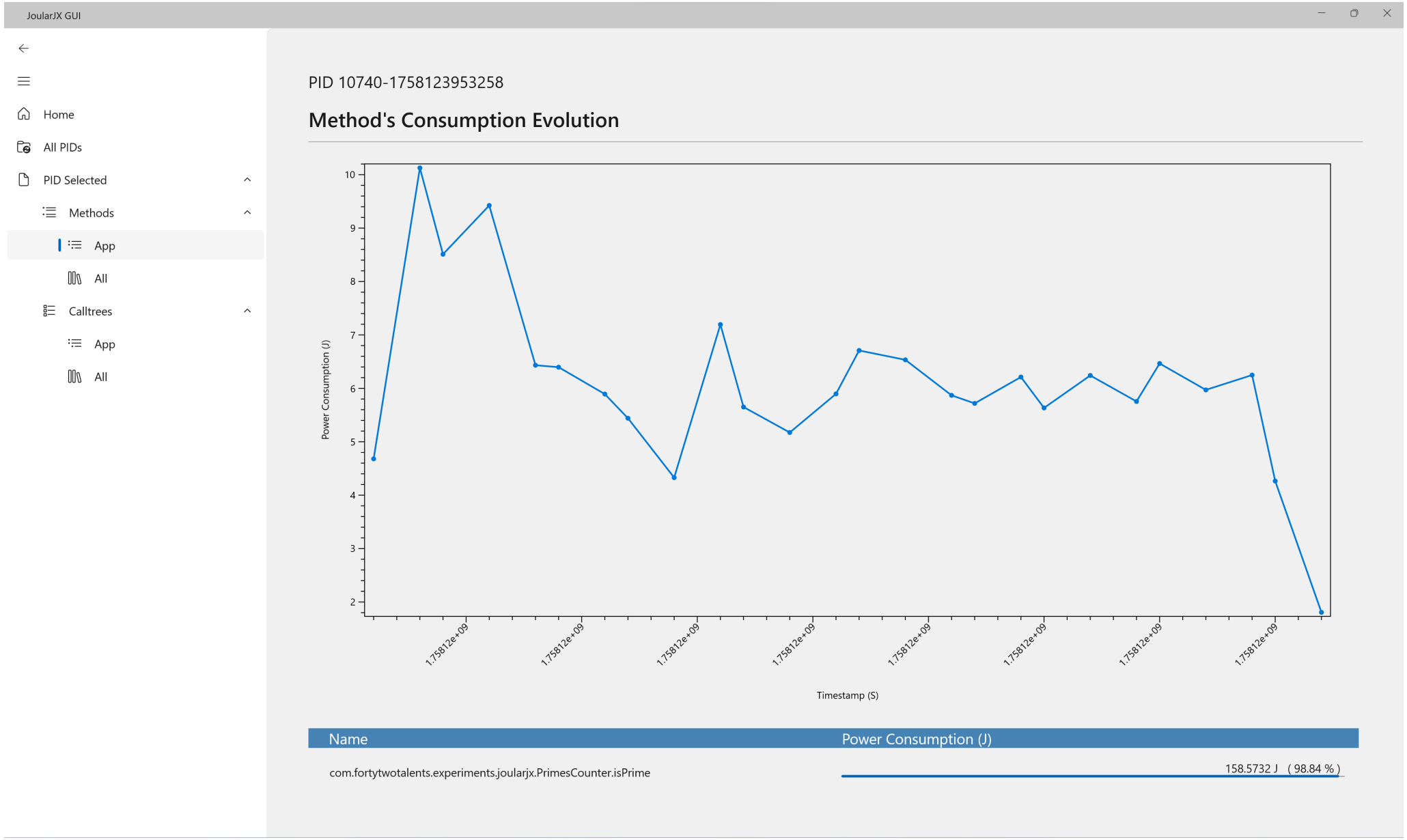
- **Performance**: latency, throughput, response time
- **Energy**: joules per transaction, useful work per joule
- Faster ≠ always greener
- Tools:
  - **JoularJX**, **PowerJoular**, Power API
  - **JMeter**, **Gatling** for load
  - **JFR**, **async-profiler** for hotspots

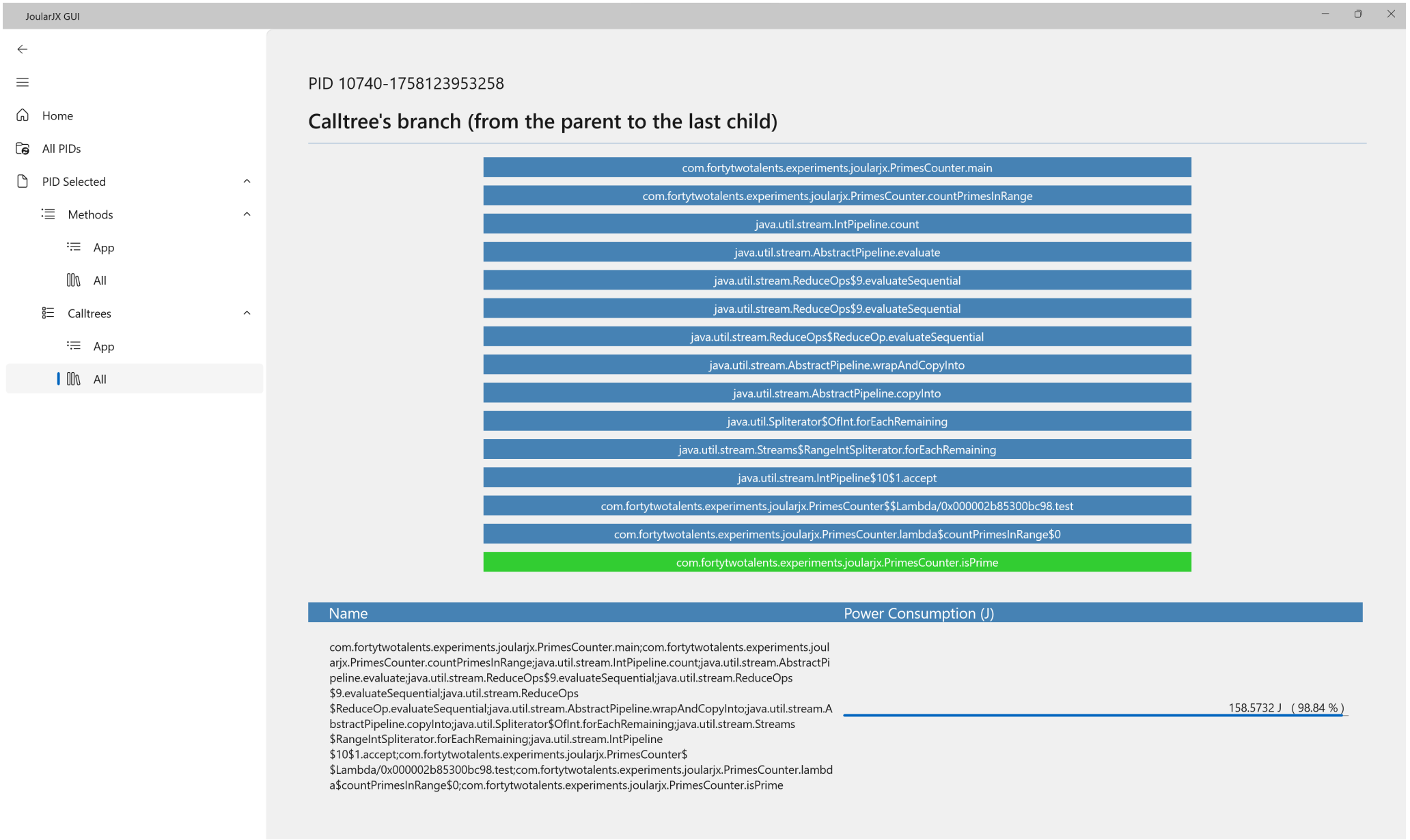
Joular  JarFX

See https://github.com/joular/joularjx


JoularJX GUI
Home
All PIDs
PID Selected
Methods
App
All
Calltrees
App
All
PID 39456-1758131891246
App Methods Power Consumption
Search methods...
Methods
15 method(s) found
Name
Power Consumption
org.springframework.samples.petclinic.PetClinicApplication.main
65.901 J (51.63 %)
org.springframework.samples.petclinic.owner.OwnerController.findPaginatedForOwners
10.9437 J (8.57 %)
org.springframework.samples.petclinic.owner.OwnerController.findOwner
10.7809 J (8.45 %)
org.springframework.samples.petclinic.owner.VisitController.processNewVisitForm
8.6048 J (6.74 %)
org.springframework.samples.petclinic.owner.VisitController.loadPetWithVisit
8.0416 J (6.3 %)
org.springframework.samples.petclinic.vet.VetController.findPaginated
6.1336 J (4.81 %)
org.springframework.samples.petclinic.owner.OwnerController.showOwner
5.8061 J (4.55 %)
org.springframework.samples.petclinic.owner.PetTypeFormatter.parse
4.2024 J (3.29 %)
org.springframework.samples.petclinic.owner.PetController.findOwner
2.7595 J (2.16 %)
org.springframework.samples.petclinic.owner.PetController.processCreationForm
1.9986 J (1.57 %)
org.springframework.samples.petclinic.owner.Visit.<init>
1.7806 J (1.4 %)
org.springframework.samples.petclinic.owner.PetController.populatePetTypes
0.4153 J (0.33 %)
org.springframework.samples.petclinic.system.WebConfiguration$$SpringCGLIB$$0.CGL
0.2021 J (0.16 %)
org.springframework.samples.petclinic.system.CacheConfiguration.petclinicCacheConfig
0.0496 J (0.04 %)


JoularJX GUI
Home
All PIDs
PID Selected
Methods
App
All
Calltrees
App
All
PID 10740-1758123953258
App Methods Power Consumption
Search methods...
Methods
2 method(s) found
Name
Power Consumption
com.fortytwotalents.experiments.joularjx.PrimesCounter.isPrime
158.5732 J (98.84 %)
com.fortytwotalents.experiments.joularjx.PrimesCounter.countPrimesInRange
1.8575 J (1.16 %)


JoularJX GUI
Home
All PIDs
PID Selected
Methods
App
All
Calltrees
App
All
PID 10740-1758123953258
Method's Consumption Evolution
Power Consumption (J)
Timestamp (S)
Name
Power Consumption (J)
com.fortytwotalents.experiments.joularjx.PrimesCounter.isPrime
158.5732 J (98.84 %)


JoularJX GUI
Home
All PIDs
PID Selected
Methods
App
All
Calltrees
App
All
PID 10740-1758123953258
Calltree's branch (from the parent to the last child)
com.fortytwotalents.experiments.joularjx.PrimesCounter.main
com.fortytwotalents.experiments.joularjx.PrimesCounter.countPrimesInRange
java.util.stream.IntPipeline.count
java.util.stream.AbstractPipeline.evaluate
java.util.stream.ReduceOps$9.evaluateSequential
java.util.stream.ReduceOps$9.evaluateSequential
java.util.stream.ReduceOps$ReduceOp.evaluateSequential
java.util.stream.AbstractPipeline.wrapAndCopyInto
java.util.stream.AbstractPipeline.copyInto
java.util.Spliterator$OfInt.forEachRemaining
java.util.stream.Streams$RangeIntSpliterator.forEachRemaining
java.util.stream.IntPipeline$10$1.accept
com.fortytwotalents.experiments.joularjx.PrimesCounter$$Lambda/0x000002b85300bc98.test
com.fortytwotalents.experiments.joularjx.PrimesCounter.lambda$countPrimesInRange$0
com.fortytwotalents.experiments.joularjx.PrimesCounter.isPrime
Name
Power Consumption (J)
com.fortytwotalents.experiments.joularjx.PrimesCounter.main;com.fortytwotalents.experiments.joularjx.PrimesCounter.countPrimesInRange;java.util.stream.IntPipeline.count;java.util.stream.AbstractPipeline.evaluate;java.util.stream.ReduceOps$9.evaluateSequential;java.util.stream.ReduceOps$9.evaluateSequential;java.util.stream.ReduceOps$ReduceOp.evaluateSequential;java.util.stream.AbstractPipeline.wrapAndCopyInto;java.util.stream.AbstractPipeline.copyInto;java.util.Spliterator$OfInt.forEachRemaining;java.util.stream.Streams$RangeIntSpliterator.forEachRemaining;java.util.stream.IntPipeline$10$1.accept;com.fortytwotalents.experiments.joularjx.PrimesCounter$$Lambda/0x000002b85300bc98.test;com.fortytwotalents.experiments.joularjx.PrimesCounter.lambda$countPrimesInRange$0;com.fortytwotalents.experiments.joularjx.PrimesCounter.isPrime
158.5732 J (98.84 %)

Using Spring Boot Wisely

- **Persistence**
  - Use projections or native queries when appropriate
  - Avoid N+1 → batch fetching, `JOIN FETCH`
- **Config**
  - HikariCP for pooling
  - Spring Cache for hot paths
  - Profile queries with **Hibernate Statistics** & **Actuator**
- **Lean builds**
  - Only needed starters/dependencies

APIs with Less Overhead

- Send less: DTOs, pagination, filtering
- Use Gzip/Brotli, HTTP/2
- REST: simple, cache-friendly (ETag)
- GraphQL: fewer roundtrips, watch N+1 (use DataLoader)
- Cache smartly: headers, CDN, gateway cache
- Serialisation: JSON external, **Protobuf/Avro** internal

Tune the JVM for Efficiency

- Right-size heap & GC (G1/ZGC)
- **JIT:** tiered compilation
- Use **CDS/AppCDS** or **AOT/AOT Cache** to cut warm-up
- Avoid oversized thread pools
- Java ≥10 respects cgroups (Containers)
- Profile with JFR & flame graphs

> Spring tip: `spring.main.lazy-initialization=true`

Infrastructure Choices Matter

- **Regions/providers vary** in grid carbon intensity
- Target low **power and water usage efficient** (PUE/WUE) data centres
- Idle servers waste → autoscale before buying more
- Containers/serverless for spiky loads

> Place workloads in cleaner regions

Scale with Care

- Myth: “Add more servers” → Reality: fix waste first
- Optimise queries, payloads, caches, JVM
- Right-size CPU/mem to load
- Autoscale with real thresholds

Boot advantage: fast start → better autoscaling

People & Impact

Developers as Change Agents

- Every design decision has **energy cost**
- **Feedback matters** – real-time or dashboard metrics
- Studies show **double-digit % savings** when teams see energy impact
- Use **gamification, goals, social comparison** to drive culture

Shift from ***“Does it work?”*** → ***“Does it work efficiently?”***

Conclusion

Spring Boot

Spring Data

Spring for GraphQL

Spring Framework

Spring Modulith

What to Remember

- **Green coding = better coding**
- Benefits: **performance, stability, cost savings**
- Cycle: **Measure → Optimise → Repeat**

> Spring Boot 3.x + Java >21 = solid base for **green coding**

Start Tomorrow

- Add **energy + perf metrics** to CI/CD
- Fix biggest offenders: queries, payloads, pools
- Tune JVM, use lazy init
- Use greener infra: regions, providers, autoscale

> Treat **sustainability as a quality attribute**

Every optimised query,
every leaner service,
every conscious design choice
moves us towards better
and greener software ...

Green Coding with Spring Boot

Sustainability as a Path to Better Software

Patrick Baumgartner - 42talents GmbH, Zürich, Switzerland

References

- Stocker, M. Carbon Emissions in ICT
- Stocker, M. Evolution of Energy Usage in Spring Boot
- Noureddine, A. JoularJX
- Ruch, J. Towards Greener Software
- Zimmermann, O. Growing Green Software
- Zaragoza, T. et al. A systematic mapping study on software-based feedback for energy consumption
- ...and many more!