



@ GraaIVM Summit 2026

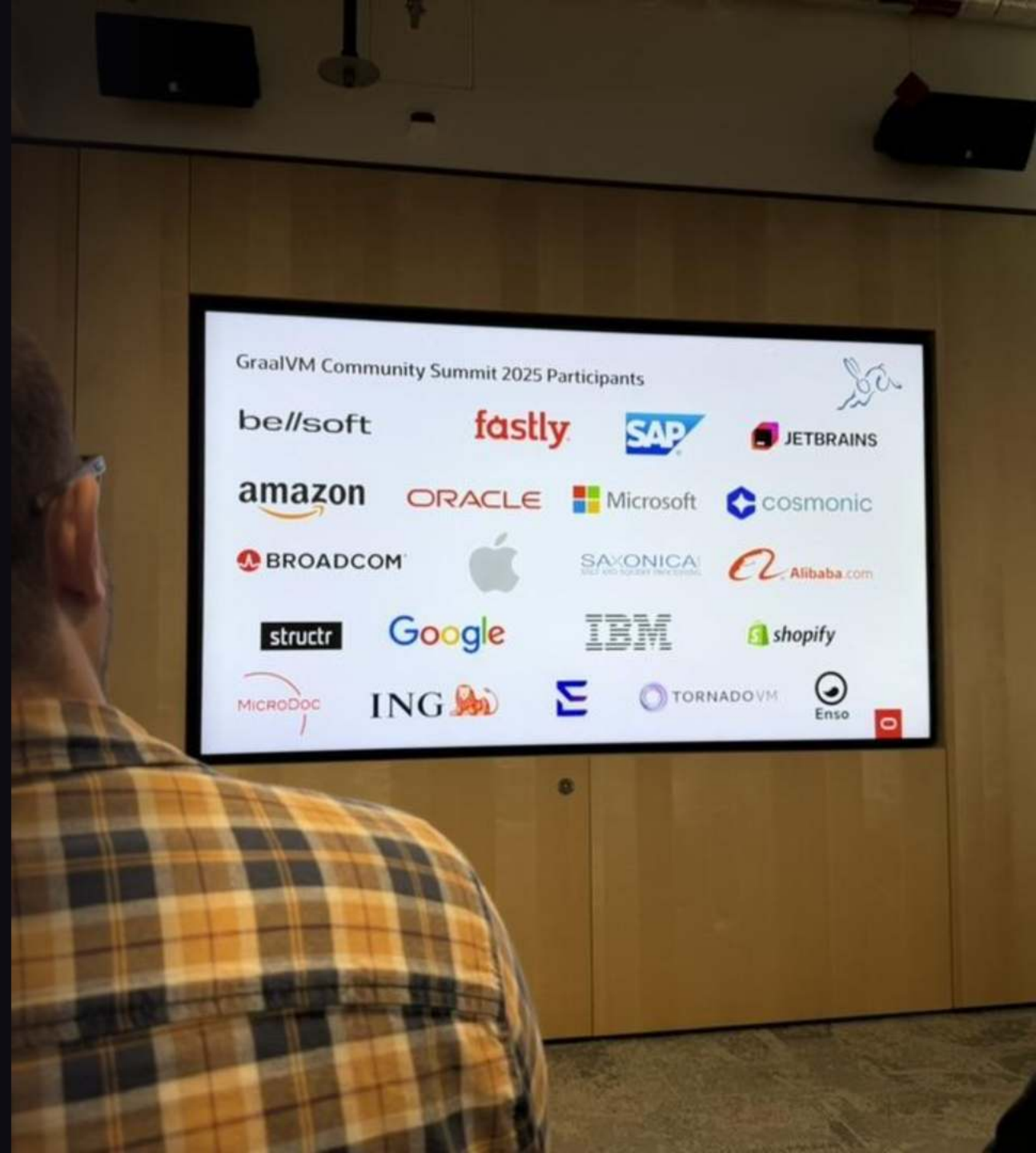
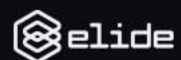
What a time to be alive!



HELLO, GRAALVM SUMMIT

Hi there!

We're Elide. We build stuff
for **software engineers**
with GraalVM.



STUFF WE BUILD

A high-performance, multi-language runtime + toolchain

Native **javac** and **kotlinc** + full JDK toolchain

20x **35x**

faster javac

faster kotlinc

Fetchers

Maven + NPM

Formatters

Google Java
Format + ktfmt

Runners

JavaScript,
TypeScript,
Python, JVM

Test runners

Coverage,
profiling, all
languages

THEN AND NOW

Elide, 1 year on

2025 summit

1.0.0-beta11-rc5

Disk footprint 600MB+

node-api ~30%

wpt-wintertc ~40%

cpython-core ~60%

java-jtreg ~99%



2026 summit

1.6.0 stable

Disk footprint <300MB

node-api 81%

wpt-wintertc 95%

cpython-core 90%

java-jtreg 100%

THEN AND NOW • ONE YEAR

JavaScript: ~21× faster start, ~12× throughput

COLD-START TIME

1.4ms

1.6.0 stable, 2026

`JavaScript · console.log(crypto.randomUUID())`

was 30ms a year ago

REQUESTS PER SECOND

10M+

1.6.0 stable, 2026

`JavaScript server · export default { fetch }`

was 800K a year ago

ELIDE 1.6.0 · JAVASCRIPT

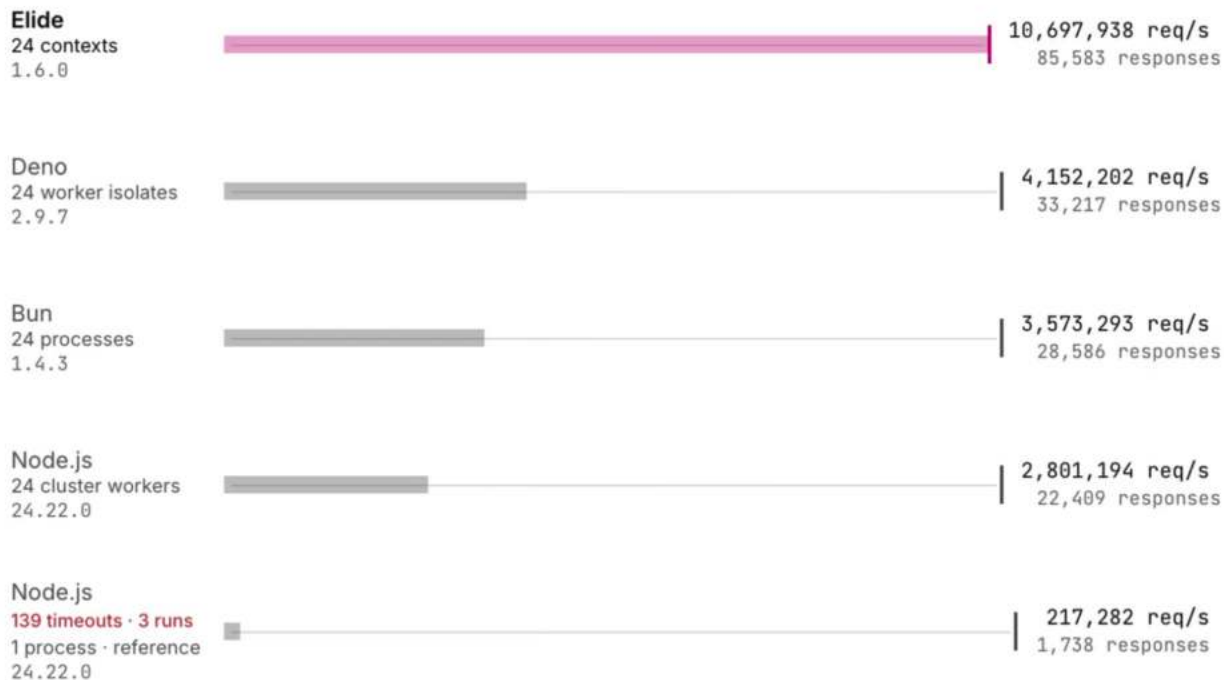
Faster to serve

10.7M

req/s · pipelined HTTP · 24 contexts

vs. Deno 4.2M, Bun 3.6M, Node.js 2.8M on the same
24 workers. Each dot is 500 responses.

Pipelined HTTP serving — measured configurations

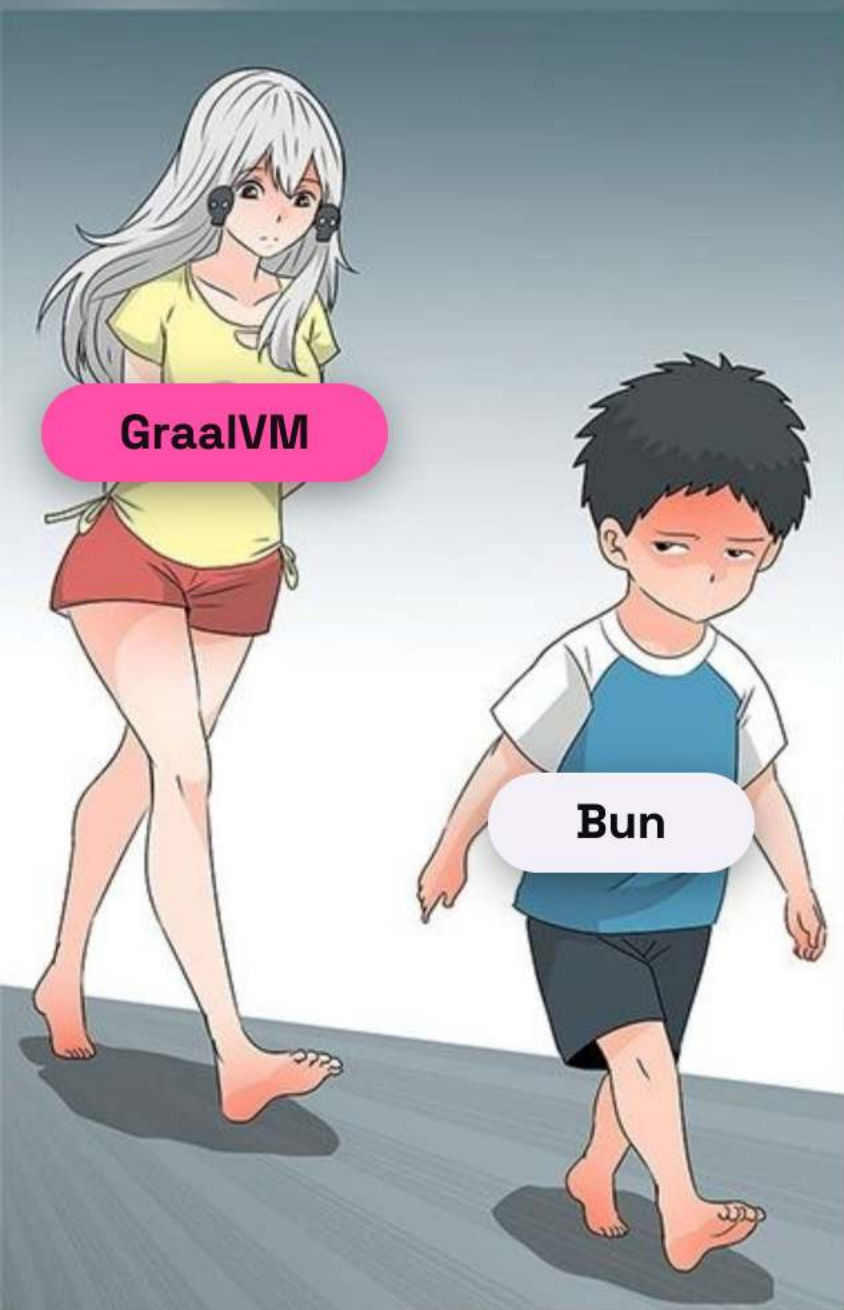


Finished: 8.0 ms measured

Speed 2×

Repeat

Higher is better. Each dot is 500 responses. Slowed 500×: 1 ms measured takes 0.5 s here. Dots cross at the same speed in every lane; only how often they leave differs.



TODAY

What I'd like to show you

01

Madura

It's like **cargo check**, but it's for Java. With one cool twist.

02

Bemo

It's a Netty transport. Best-of-breed Rust libs. Works on JVM and SVM.

03

Bali

It's a new JDK, built on top of Crema and Ristretto. With several cool twists.

04

Komodo

It's a new Kotlin, built natively on Truffle. With several cool twists.

UPSTREAM AND BEYOND

Contributions & patches to get there

GraalVM pull requests

#12796 macOS source-level debugging

#13018 Layers on macOS, Windows,
and aarch64

#12865 Cosmopolitan Libc

In our fork

PLT/GOT + JDWP with Truffle JIT

Virtual threads with Truffle JIT

AOT parse of Truffle guest code

AOT PE of Truffle guest code

Keep unused pre-initialized langs dormant

70

pull requests

382

upstream commits

31

release tags

42

issues tracked

Madura

elide-dev/**madura**

Smol, fast, universal, Java toolchain



Madura

Kotlin Island

(10,196KM AWAY)

Java Sea

Madura Island

WEST JAVA

CENTRAL JAVA

EAST JAVA

BALI

Lombok



MADURA

Benchmark: **javac**

22.5x

faster than stock javac, same flags

Benchmark 1: javac ...

⌵ Performing warmup runs



ETA 00:00:00

MADURA

Like cargo check, but for Java

```
$ madura
```

Compile Java code

Identical flags to javac. Native javac via native-image.

```
$ madura check
```

Check Java code

Only javac's analysis phase runs. Code-gen is skipped, like cargo check.

39.5MB

**Self-contained,
extremely minimal**

Zip download, no dependencies.
No JAVA_HOME needed (but can use one for API visibility).

MADURA

Wait, one release asset?

▼ Assets 5

 madura-1.2.2-cosmo-universal.zip	sha256:...		1 	39.5 MB	2 weeks ago
 SHA256SUMS	sha256:...		1 	99 Bytes	2 weeks ago
 Source code (zip)					2 weeks ago
 Source code (tar.gz)					2 weeks ago
 Release attestation (json)					2 weeks ago

???

Write once, build once, run everywhere

Cosmopolitan Libc

One target: every operating system, every architecture.

Actually Portable Executable

One binary format. Works across 6 OSes and self-dispatches multi-arch.

```
$ native-image --libc=cosmo
```

It works! Build once, test your native code on every OS and arch.



actually portable executable



One binary per OS and arch



One APE binary, every OS and arch

Bemo

elide-dev/**bemo**

Netty fortified with Rust



Native Image C API beats JNI and Panama

A drop-in Netty transport

On the JVM, via FFM (Panama): works great, and improves performance over JNI.

On Native Image, via the NI C API: radically faster. Bemo builds as a **static object linked into the final binary**, not a shared library embedded in resources.

```
libbemo.{so,dylib} via FFM ·  
static NI C API linked into  
the binary
```

It's really fast

18M **15M**

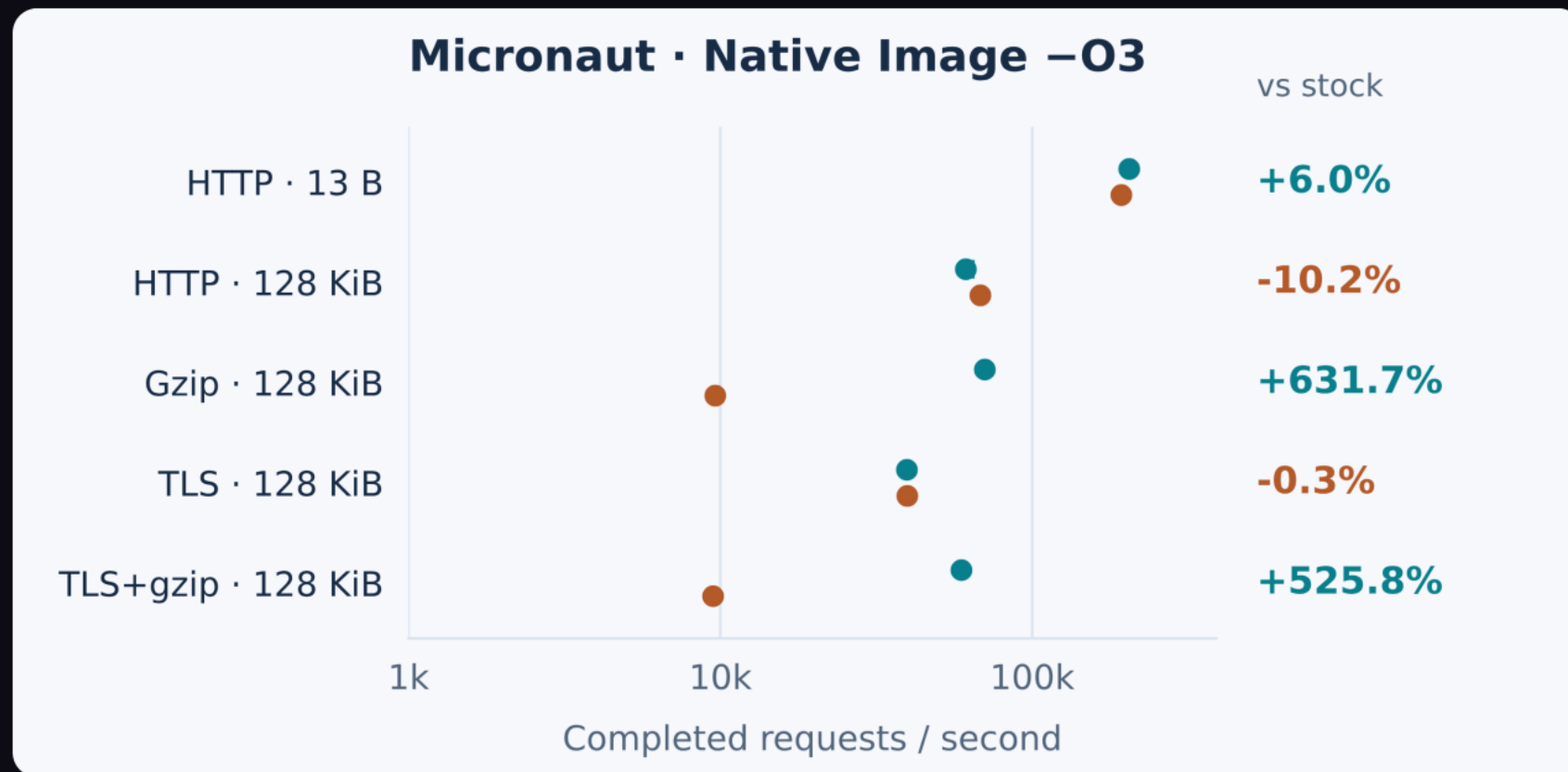
RPS, plain HTTP RPS, TLS

Sustained with Elide. Accelerated HTTP parsing/encoding, bulk crypto and compression, NI C API.

Replaces JNI-loaded libs

```
crypto  
AWS-LC-RS, the core of ACCP  
  
compression  
zlib-rs  
  
transport  
compio
```

Micronaut on Native Image, Bemo vs. stock



+632%

gzip · 128 KiB

+526%

TLS+gzip · 128 KiB

Bemo: io_uring / zlib-rs / Rustls.
Stock: Netty epoll / JDK gzip /
tcnative BoringSSL.

03 • A NEW JDK

Bali

Built on top of Crema and Ristretto, with several cool twists.



BALI

A drop-in JDK 25

Compatible

Passes the TCK (via upstream) and jtreg.

Use it as your build-time
JAVA_HOME: Gradle and Maven
get a lot faster.

Shenandoah GC:
likely default

Runs Crema + Ristretto

5–15×

faster on short-running tasks

2–5×

less memory usage

Only ~2× slower on soak tests.

Fortified with Rust

Crypto backed by ACCP and aws-lc-rs

Faster primitives: crc32c (SIMD),
libz (flake)

Faster transport via io_uring

All optional: targeted substitutions

04 • A NEW KOTLIN

Komodo

Built natively on Truffle, with
several cool twists.



Kotlin on Truffle

A Kotlin-native VM

Feature parity with Kotlin 2.5.0.

Standalone implementation, with kotlinc as a behavior oracle.

Not to be confused with Kotlin/Native as in KMP.

Strong A1-first implementation

Stiff budgets for allocation and instruction count.

Continuous JIT bailout testing with **-H:CompileFailAction=Throw**.

All language features, all of stdlib, all of KotlinX.

Many benefits from Graal

JVM interop via Crema + Polyglot

Interpreted: owned IR → Bytecode DSL

AOT: Truffle ahead-of-time + native-image

WASM: on GC or LM

Tooling: DAP, Chrome DevTools, profiling, coverage

Bali can speak Java + Kotlin



Why Kotlin on Graal can beat K2 on C2

Feature	K2 → JVM → C2	Komodo → Graal
Nullable primitives	⚠ Always boxed; EA stops at calls	✓ Primitive slot + null tag
Generic primitives	⚠ Erased, boxed on the heap	✓ Specialized per call site
Value classes	⚠ Boxed at generic / nullable edges	✓ Scalar through PE + PEA
Null safety	⚠ Erased to runtime checks	✓ Static; checked only at Java edge
Coroutines	⚠ CPS state machine, spilled locals	✓ Native frames, no spill
Polymorphic calls	⚠ 2 types, then itable dispatch	✓ Split per caller → monomorphic
Speculation	⚠ Trap limits switch it off	✓ Node-local, accumulates
Inlining budget	⚠ 35 / 325 bytecode bytes	✓ PE over the whole hot tree
Reflection in core	⚠ lazy, coroutines, atomicfu	✓ None

KOMODO

Where it stands

Implemented

- ✓ All language features
- ✓ All of stdlib
- ✓ All of KotlinX
- ✓ Native coroutines

Continuously tested

- ✓ Ktor
- ✓ Spring Boot
- ✓ Micronaut

Kotlin production-readiness probes

JVM oracle · uncached / cached / adaptive · archived owned IR

■ Supported: 164

Production-oriented workloads: 30 / 30 supported



Language probes (stable Kotlin): 126 / 126 supported

Faster to first result, lighter all the way

5.9_x

faster to first result

median 3.7 ms vs 22.2 ms, 37 workloads

11_x

less CPU to first result

user + system time, same runs

1.8_x

work done in the first 100 ms

median over workloads, from process launch

2.2_x

less peak memory

median per workload; less on 22 of 37

53_x

faster cold compile

32 ms vs 1,730 ms

37 / 37

kernels at C2 parity or better

Komodo native JIT within 1.25_x of C2

Advancing the Pareto frontier for Kotlin



fibonacci: 0.01× C2

12.1 ns against 1,048.0 ns per call; 8 B against 0 B.

sealed-when: 0.01× C2

12.3 ns against 990.3 ns per call; 8 B against 18,432 B.

sequence-ops: 0.01× C2

26.0 ns against 1,815.7 ns per call; 8 B against 13,872 B.

LIVE DEMO

Explorer

- elide-android-sample
 - .dev
 - .tools/bin
 - elide
 - .vscode
 - extensions.json
 - settings.json
 - tasks.json
 - app
 - .dev
 - src/main
 - kotlin/sample/app
 - MainActivity.kt
 - PreviewGallery.kt
 - res
 - AndroidManifest.xml
 - elide.pkl
 - components
 - .dev
 - src/main
 - kotlin/sample/components
 - JourneyCard.kt
 - res
 - AndroidManifest.xml
 - elide.pkl
 - .gitignore
 - elide.pkl
 - README.md
 - workspace.json
- Outline
- Timeline
- Zip Explorer

```

components > src > main > kotlin > sample > components > JourneyCard.kt > JourneyCard
19 data class Journey(val destination: String, val detail: String, val stops: Int)
20
21 class JourneyProvider : PreviewParameterProvider<Journey> {
22     override val values = sequenceOf(
23         Journey(destination = "Coastal express", detail = "Ocean views all the way home",
24             Journey(destination = "Mountain weekend", detail = "A longer title that wraps on compact screen",
25             Journey(destination = "Stay local", detail = "Your next adventure starts nearby", stops = 0),
26     )
27 }
28
29 @Composable
30 fun JourneyCard(journey: Journey, modifier: Modifier = Modifier) {
31     val dark = isSystemInDarkTheme()
32     val ink = if (dark) Color(0xFFE9F1EF) else Color(0xFF16332D)
33     Column(modifier.fillMaxWidth().background(color = if (dark) Color(0xFF16332D) else Color(0xFFE9F1EF))) {
34         BasicText("Hello Uber!")
35         BasicText("They're inspectable too!")
36         errors are easy to diag
37         BasicText(stringResource(id = R.string.journey_label), style = TextStyle(color = if (dark) Color(0xFF16332D) else Color(0xFFE9F1EF)))
38         Spacer(Modifier.height(12.dp))
39
40         BasicText(journey.destination, style = TextStyle(color = ink, fontSize = 26.sp, fontWeight = FontWeight.Bold))
41         Spacer(Modifier.height(64.dp))
42
43         BasicText(journey.detail, style = TextStyle(color = ink, fontSize = 16.sp))
44         Spacer(Modifier.height(20.dp))
45
46         BasicText(stringResource(id = R.string.journey_stops, formatArgs = journey.stops), style = TextStyle(color = ink, fontSize = 16.sp))
47     }
48 }
49
50 // Open this library file from the workspace root: app supplies linked resources.
51 @Preview(name = "Library resource context", widthDp = 340, showBackground = true)
52 @Composable
53 fun LibraryCardPreview() {
54     JourneyCard(Journey(destination = "Shared components", detail = "Rendered using the consuming app's resources", stops = 2))
55 }
56
57 @Preview(name = "Library sample data", widthDp = 300)
58 @Composable
59 fun LibraryDataPreview(@PreviewParameter(provider = JourneyProvider::class, limit = 2) journey: Journey) {

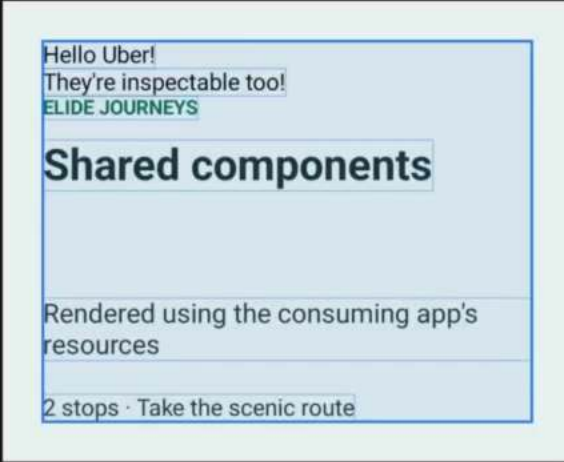
```

Compose: JourneyCard.kt

Compose previews 3 Rebuild Fit Inspect

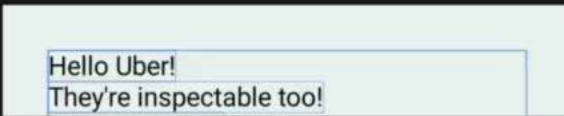
Updating preview... LAST GOOD PREVIEW Build log

Library resource context
sample.components.JourneyCardKt.LibraryCardPreview



Column [Open source ↗](#)
Waiting for fresh source locations - preview retained
Column — JourneyCard.kt:33 ⌵ ↑ Parent

Library sample data
sample.components.JourneyCardKt.LibraryDataPreview





Thank you

find us at elide.dev