Working through JDK 25: The Why

Speaker: Vaibhav Choudhary

JVM Engineer, Bangalore JUG Lead, OpenJDK Author

Why Upgrade to JDK 25?

⚓The Next LTS Milestone

JDK 25 is the next Long-Term Support (LTS) release following JDK 21. It provides a stable, supported platform for enterprise applications for years to come.

Consolidated Innovation

It consolidates major innovations introduced in JDK 22, 23, and 24, delivering them in a production-ready form. This includes finalized features like FFM and Virtual Threads improvements.

Gold Standard Performance

SpecJBB2015 Benchmark Results (Changing JDK at runtime)

+113%
Critical jOPS
vs JDK 8

+31%
Max jOPS
vs JDK 17

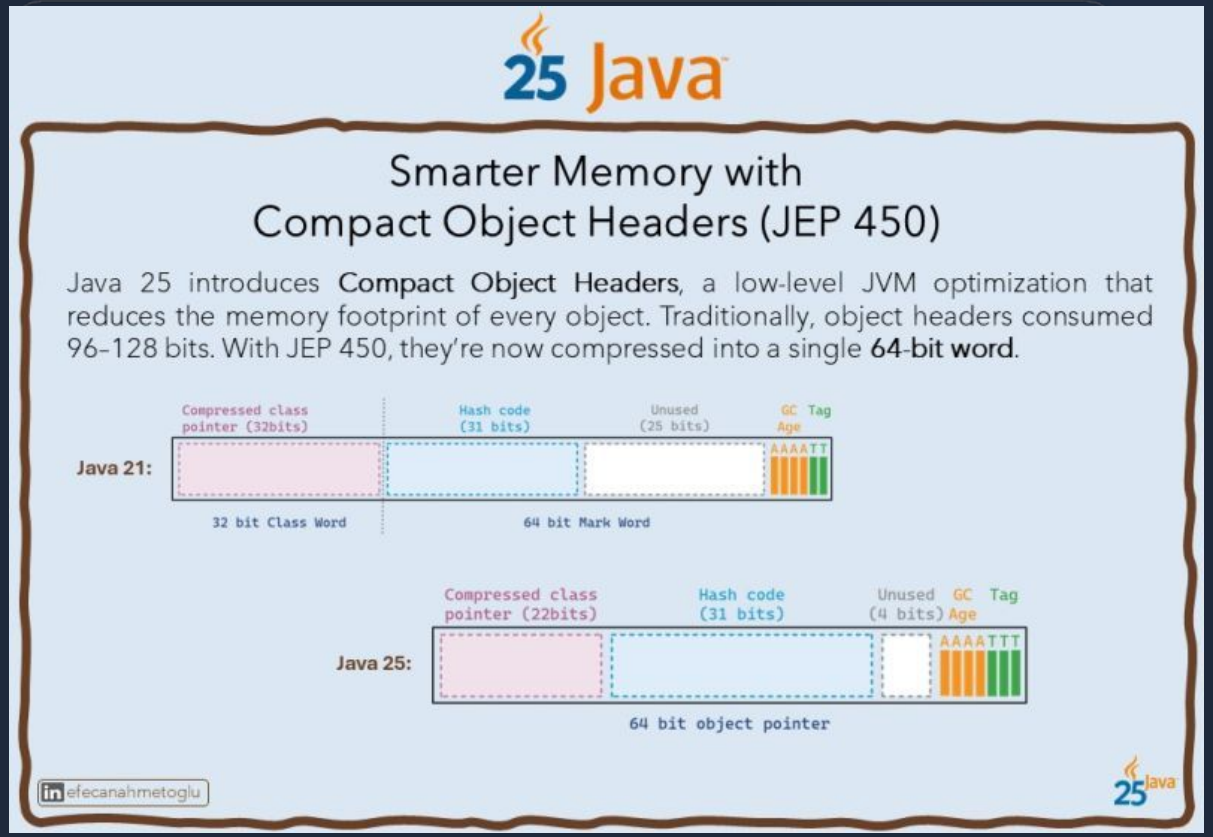
+10%
Critical jOPS
vs JDK 21

Compact Object Headers (JEP 450)

Efficiency at Scale

Traditionally, every Java object carries a metadata header. JDK 25 shrinks this, reducing per-object overhead significantly.

- **~10% Reduction:** Lower memory footprint observed in many applications.
- **Better Caching:** Smaller headers mean more effective use of CPU caches.
- **Denser Heaps:** Fits more objects in memory, reducing GC pressure.



Developer Experience & Safety

Virtual Threads

JEP 491: Production-ready completeness. Removes synchronization pinning issues, ensuring high throughput and fewer blocking scenarios without workarounds.

FFM API

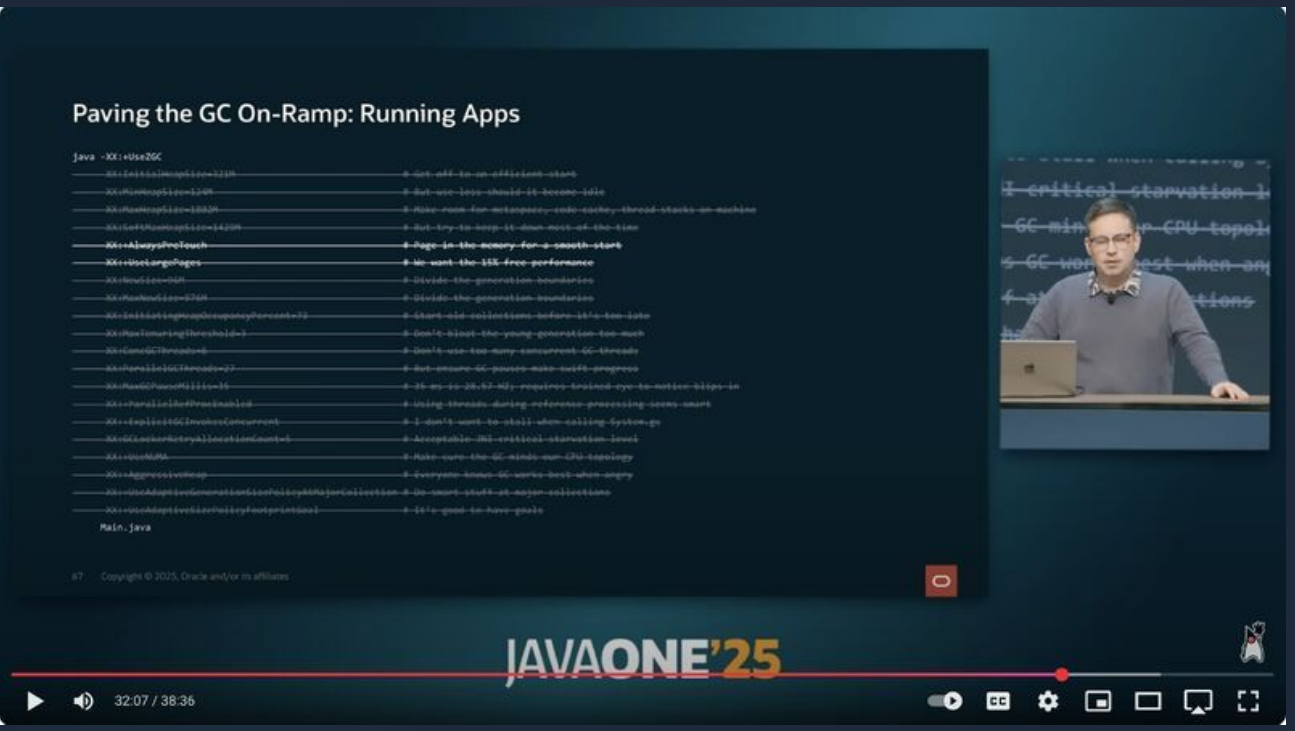
JEP 454: A modern, safer replacement for JNI. Enables high-performance native interop without boilerplate and with better safety guarantees.

G1 GC Optimizations

Late Barrier Expansion (JEP 475): Postpones write barriers to reduce JIT overhead.

Metric (Stress Test)	JDK 21	JDK 25	Improvement
Total Runtime	3830 ms	3485 ms	**~10% Faster**
C2 Compile Time	1.020 s	0.707 s	**~30% Reduced Overhead**
Execution with -Xcomp	3877 ms	3223 ms	**~17% Faster**

ZGC Optimizations



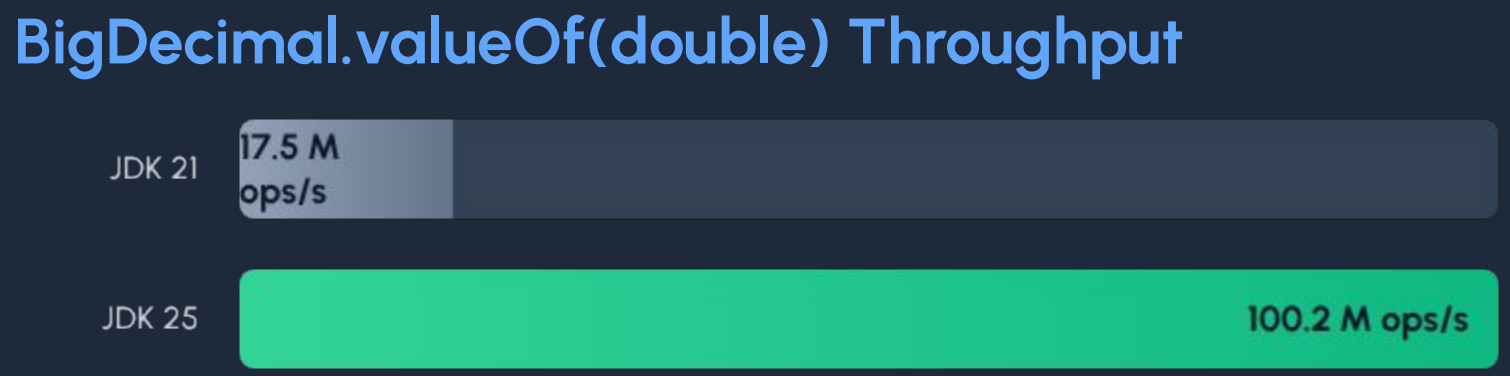
Generational ZGC Enhancements

Concurrent Remap Roots: Massive optimization for major collection phases. Micro-benchmarks show a reduction from **1.15ms to 0.03ms** (~31x speedup).

Mapped Cache: New page allocation strategy that merges adjacent memory ranges, reducing fragmentation and OOME risks in long-running apps.

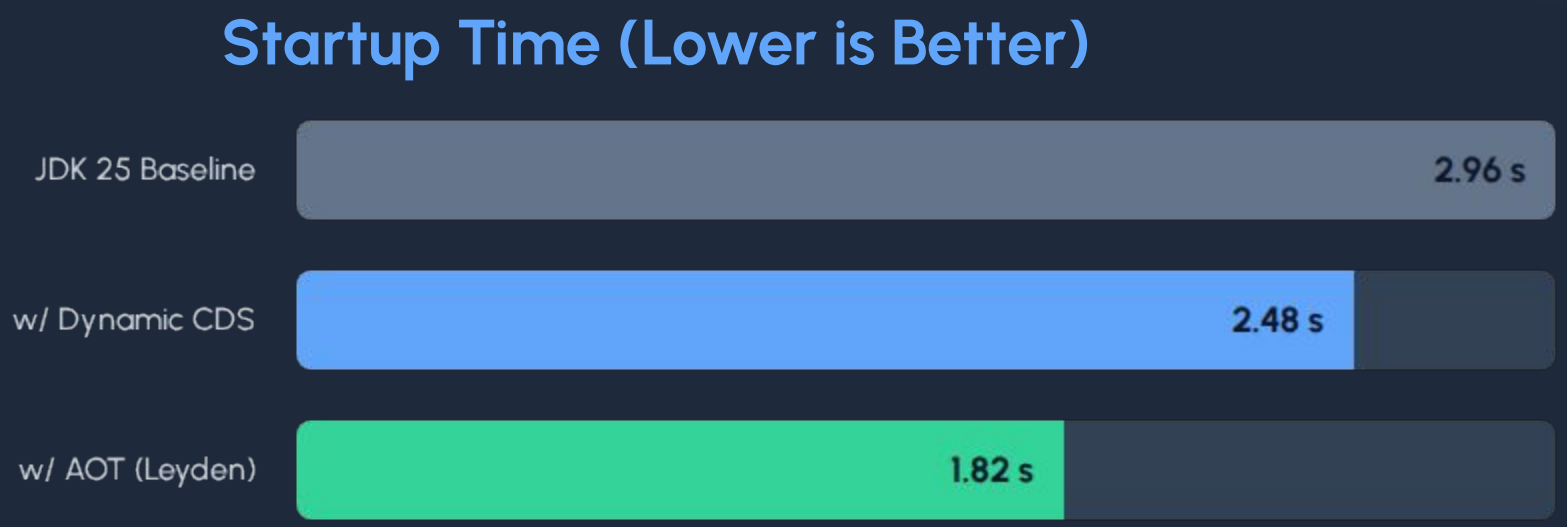
Autotuning: Fully autonomous configuration for optimal performance.

Runtime & Library Gains



Math Intrinsics: Math.min / Math.max latency reduced from 210ms to ~26ms.

Startup: Project Leyden



~38.5% improvement in startup time using Ahead-of-Time (AOT) profiles.

Debugging & Observability

Enhanced JFR

JDK 25 empowers Java Flight Recorder with **CPU-Time Profiling (JEP 509)**. This closes a critical gap in diagnosing CPU regressions compared to previous versions.

New Events

Includes **Method Timing & Tracing** (JEP 520) and **Cooperative Sampling** (JEP 518) for more granular and accurate performance insights.

The Watch-outs

⚠Compact Header Risks

JEP 450 changes massive amounts of code around GC and Klass metadata. This creates a risk for regression in stability or native interoperability.

Action Required

Teams must prioritize rigorous regression testing around GC behavior and object-heavy workloads before production rollout.

Questions?

Thank you!