

Agenda

- Discussion
 - Cloud Native
 - GraalVM JIT
 - GraalVM AOT (native image)
- Resources → try it at home / in a sandbox
- Walk through / demo

“We need to deliver SaaS”

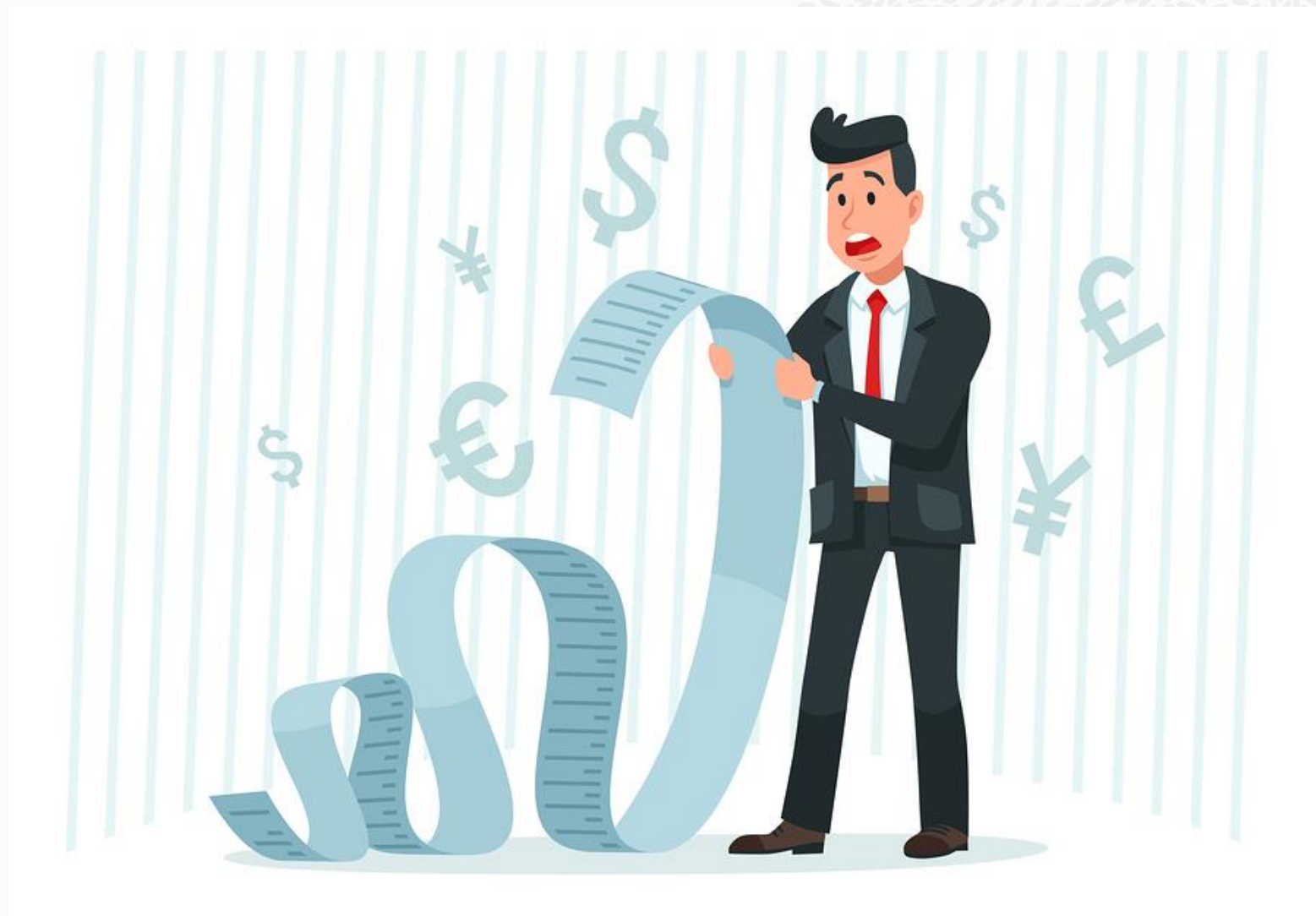


SaaS implementation Plan

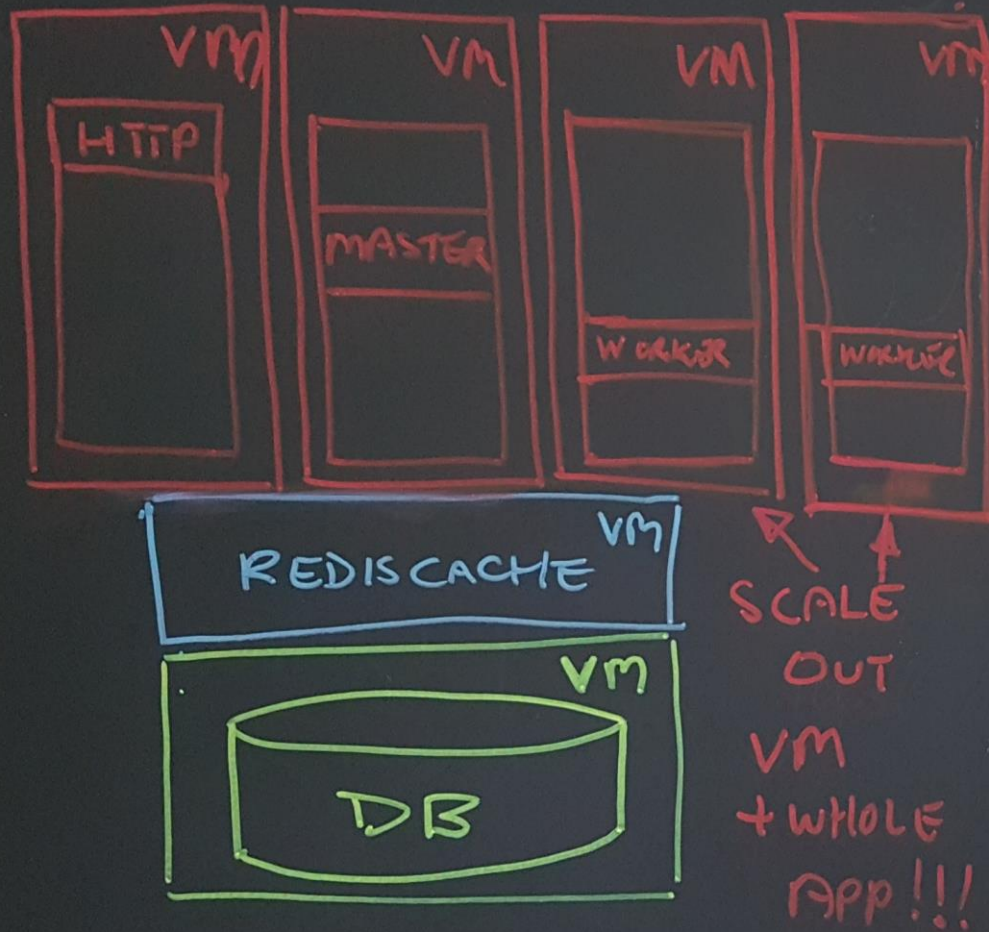
- Virtualise on – premises solution
- Host VMs in cloud
- Set up network access
- Sign up customers



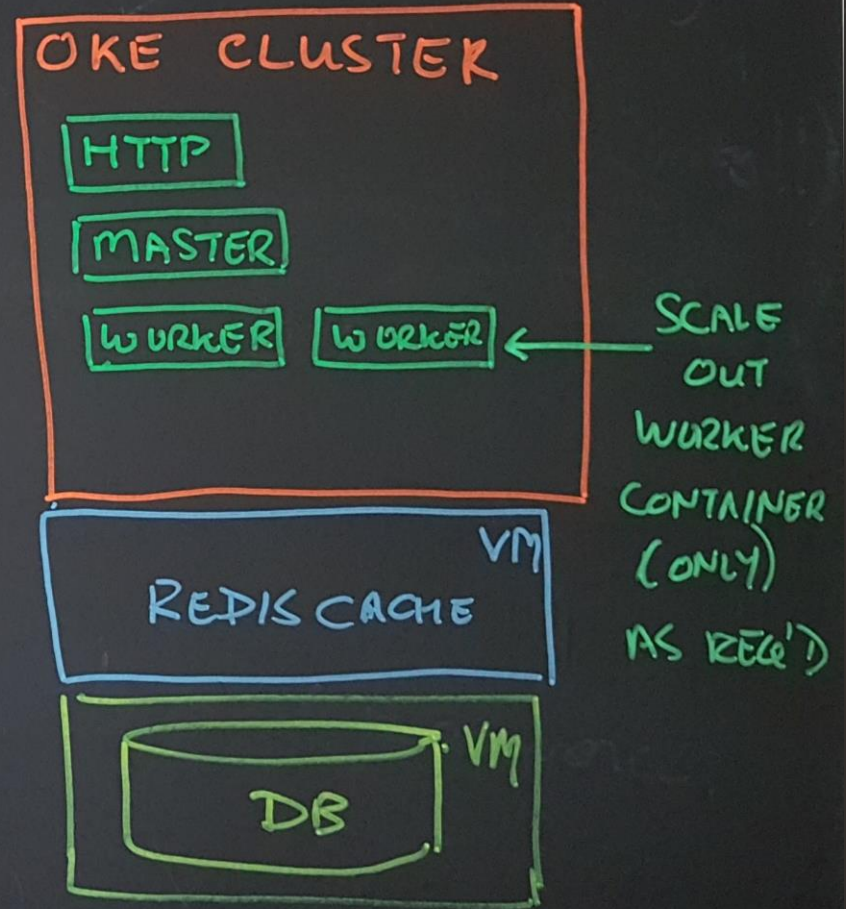
Invoiced based diagnosis of a scalability issue



Original Architecture



New Architecture



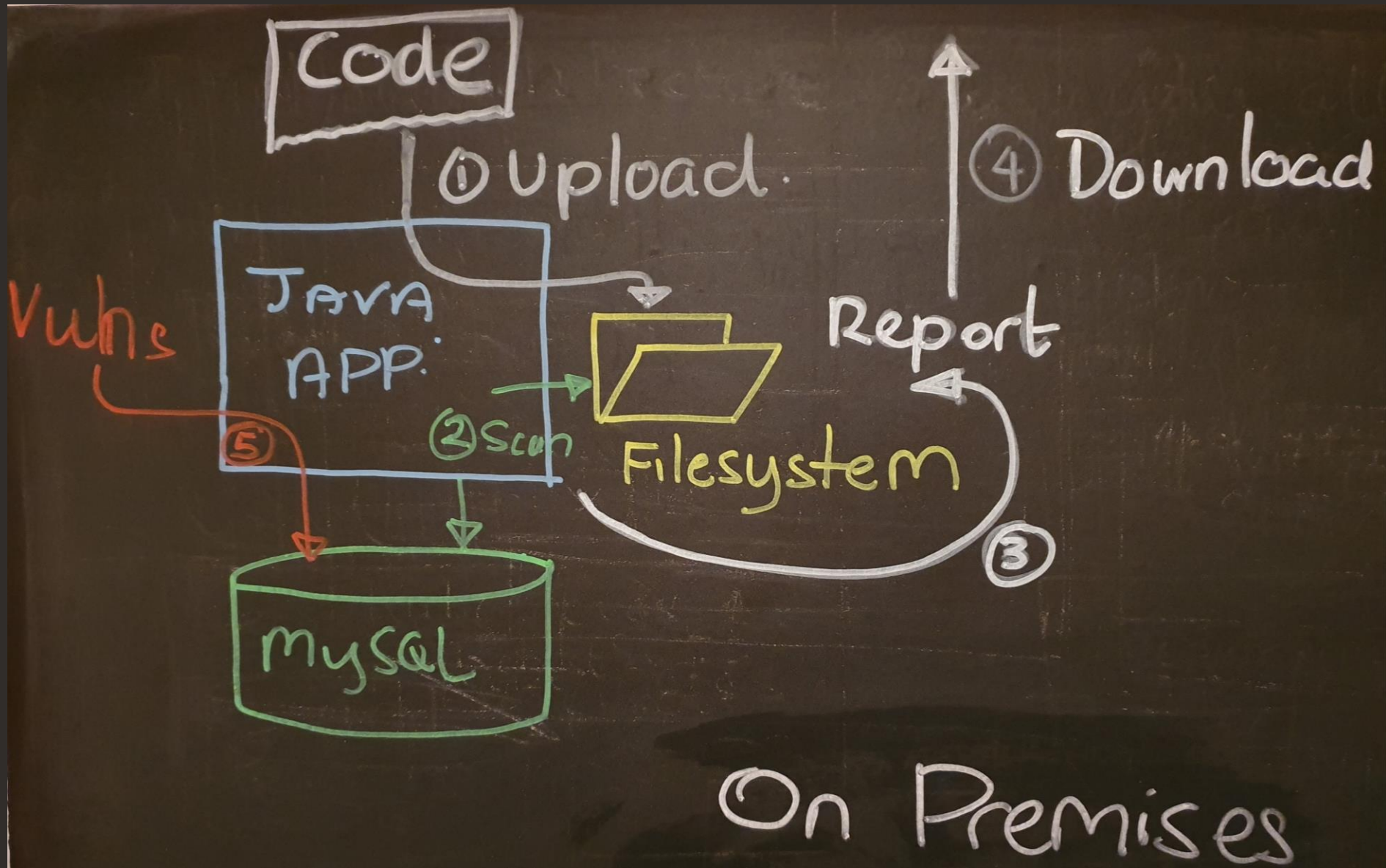
“Constructive Laziness”

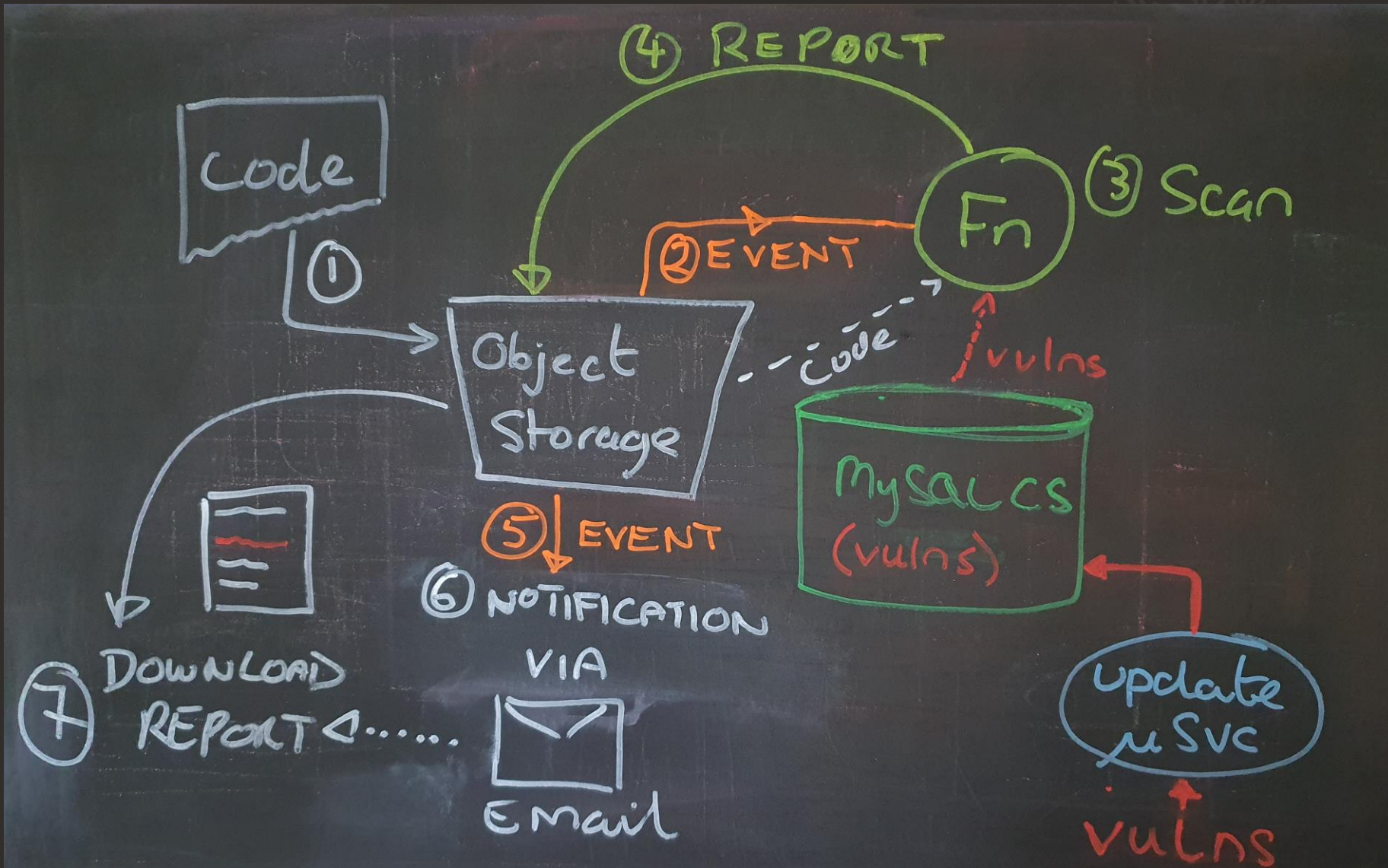


“Simplicity – the art of maximizing the amount of work not done – is essential.”

Agile Principle #10

<http://agilemanifesto.org/principles>





“Use the ~~force~~ platform”



Cloud Economics



- Minimise infrastructure costs
 - Only run
 - What you need
 - When you need it
- Minimise development costs
 - Only develop what you need to
 - Delegate to the platform when you can





CLOUD NATIVE COMPUTING FOUNDATION



CNCF Cloud Native Definition v1.0

Approved by TOC: 2018-06-11

[日本語版](#) (Japanese) | [한국어](#) (Korean) | [Deutsch](#) (German) | [Español](#) (Spanish) | (Hebrew) [עברית](#) | [中文版本](#) (Chinese) | (Arabic) [العربية](#)
[Français](#) (French) | [Polski](#) (Polish) | [Português Brasileiro](#) (Portuguese-BR) | [Português de Portugal](#) (Portuguese-PT) | [Русский](#) (Russian) |
[Bahasa Indonesia](#) (Indonesian) | [Türkçe](#) (Turkish) | [Български](#) (Bulgarian) | [ไทย](#) (Thai) | [Magyar](#) (Hungarian) | [Hindi](#) (Indian)

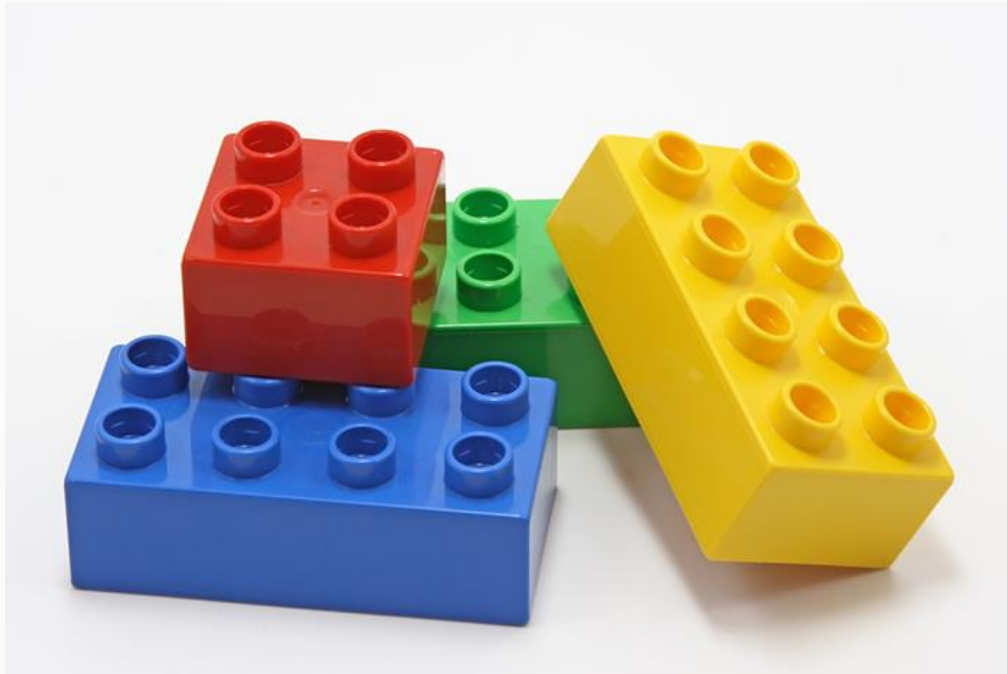
Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.

Smaller components

Serverless, Microservices



“Runtime Efficiency”

—
(doing more with less)



A decade of GraalVM research @ Oracle

**A multilingual VM can be 100% compatible with
any language, meeting or beating performance of
single language VMs**




GraalVM


graalvm.org

Waveney-slt.co.uk Bookmarks OCI-Cert Unseen University Work Chromebook Code Life


High Performance. Cloud Native. Polyglot.




Java




JavaScript




Python



Ruby



R



WASM

Download

Get Started

Oracle Developer Live: Java Innovations

←

GraalVM, Spring and Containers

TUESDAY, MAR 22 5:20pm PT

Register

Oracle Developer Live: Java Innovations


→

Accelerating Productivity with Java, GraalVM and Micronaut


TUESDAY, MAR 22 1:25pm PT

Register

Key Features



High Performance



AOT Native Image Compilation


GraaIVM

graalvm.org/downloads/

Waveney-slt.co.uk Bookmarks OCI-Cert Unseen University Work Chromebook Code Life

GraaIVM Docs Community Videos Blog Download

Download GraaIVM




GraaIVM 22.0 GraaIVM 21.3 GraaIVM 20.3 Developer Builds

GraaIVM Community 22.0

Community supported open source build

[Release notes](#)

Download




GraaIVM Enterprise 22.0


Oracle 24x7 supported commercial build

[Release notes](#)


Download



Docker Images

 **GraaIVM Community Container Images**

[Browse the GitHub Container Registry](#)

 **GraaIVM Enterprise Container Images**

[Browse the Oracle Container Registry](#)





master

63 branches 215 tags

Go to file

Code

	eregon [GR-26395] Periodic update of the trufferuby import. ...	ad19dd4 2 hours ago	🕒 69,291 commits
📁 .devcontainer	Simplify devcontainer config.		4 months ago
📁 .github	moved mx_version from graal_common.json to common.json		3 days ago
📁 ci_includes	Add javadoc under version-specific directory in graalvm-website		7 months ago
📁 compiler	[GR-41636] Fix canonicalization of conditional to truncate (GitHub #5101)		5 hours ago
📁 docs	[GR-41207] Add missing code snippet in Debug Native Executables wit...		18 days ago
📁 espresso	Group ci files in each suite		yesterday
📁 java-benchmarks	Quick fix		yesterday
📁 regex	Group ci files in each suite		yesterday
📁 sdk	[GR-41577] Remove GuardedAnnotationAccess and DirectAnnotation...		15 hours ago
📁 substratevm	[GR-41019] Always run phaseplan verification.		9 hours ago
📁 sulong	Quick fix		yesterday
📁 tools	Group ci files in each suite		yesterday
📁 truffle	Group ci files in each suite		yesterday

About

GraalVM: Run Programs Faster Anywhere 🚀

www.graalvm.org

- javascript
- ruby
- python
- c
- java
- vm
- r
- polyglot

- 📖 Readme
- 📄 View license
- 📄 Code of conduct
- ★ 17.6k stars
- 👁 467 watching
- 🍴 1.4k forks

Contributors 250



+ 239 contributors

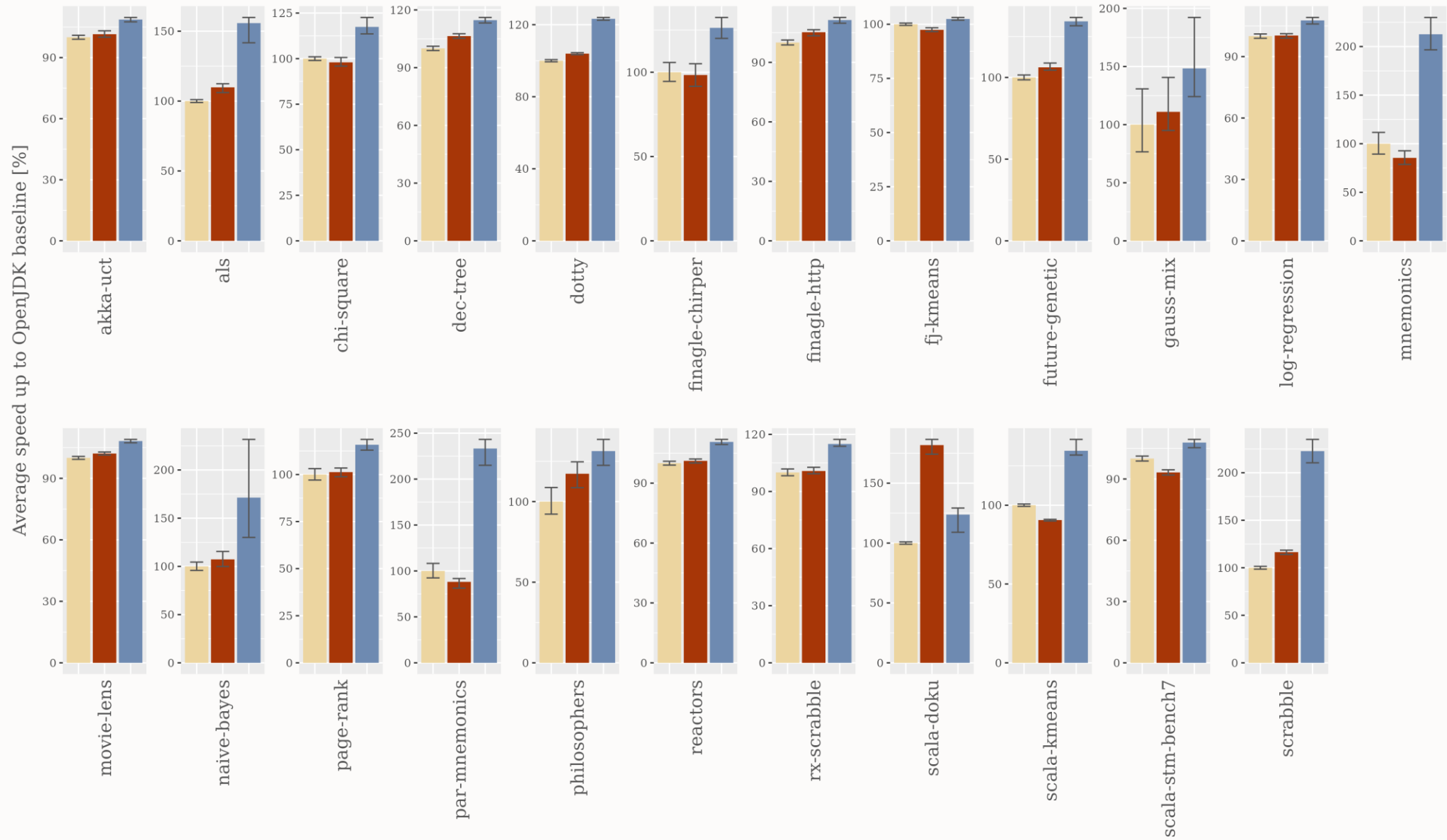
GraalVM Enterprise Edition



- Included with OCI
- Production Support
- Higher performance, more advanced than CE



Renaissance Suite (higher is better)



Most of Graal Is Developed in Java



```
package org.graalvm.compiler.phases.common;

public class ConditionalEliminationPhase extends BasePhase<CoreProviders> {
...
    protected void run(StructuredGraph graph, CoreProviders context) {
        try (DebugContext.Scope s = graph.getDebug().scope("DominatorConditionalElimination")) {
            BlockMap<List<Node>> blockToNodes = null;
            NodeMap<Block> nodeToBlock = null;
            ControlFlowGraph cfg = ControlFlowGraph.compute(graph, true, true, true, true);
            if (fullSchedule) {
                if (moveGuards && Options.MoveGuardsUpwards.getValue(graph.getOptions())) {
                    cfg.visitDominatorTree(new MoveGuardsUpwards());
                }
            }
        }
    }
...
}
```

Simpler programming model, Java ecosystem, debugging tools

Faster progress!



Most of Graal's Tooling Is Developed in Java

The screenshot displays the Graal.js IDE interface. The main window shows a control flow graph (CFG) for a method named 'silly.js'. The graph consists of several nodes connected by edges, representing the execution flow of the code. The nodes are color-coded and labeled with their IDs and names:

- 204 LoadingClass
- 313 VirtualFrameSet
- 373 VirtualFrameSet
- 454 VirtualFrames
- 455 If
- 456 ==
- 457 Begin
- 458 Begin
- 462 VirtualFrameGet
- 469 Deopt TransferToInterpreter
- 474 InstanceOf
- 475 ...
- 107 ...

The right-hand side of the IDE features a 'Filters' panel with the following settings:

- Coloring:
- Remove State:
- Probability Coloring:
- Reduce Edges:
- Remove Floating:
- Call Graph Coloring:
- Stamp Coloring:

Below the filters is a 'Stack View' section. The current node selected is '455: If'. A tooltip is visible over the 'Locate in Java Project' icon, containing the following text:

- Locate in Java Project
- Add root of sources

Oracle GraalVM Enterprise Edition



Faster



Smarter

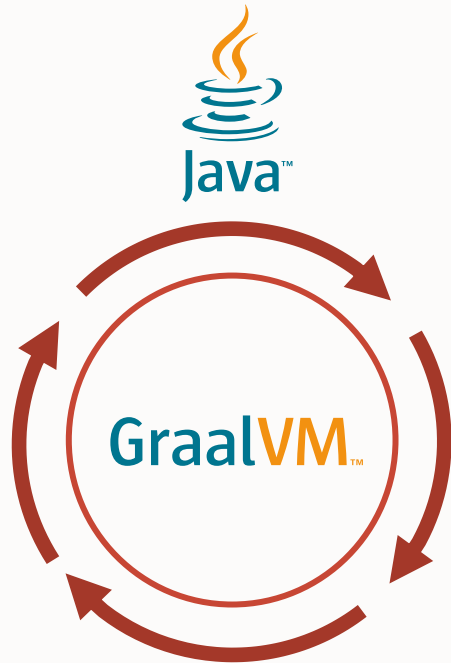


Leaner



What is GraalVM Enterprise?

A distribution of Oracle JDK with the high performance GraalVM optimizing compiler that provides significant improvements in application speed and efficiency



High-performance optimizing Just-in-Time (JIT) compiler



Ahead-of-Time (AOT) "Native Image" generator



Multi-language support for the JVM

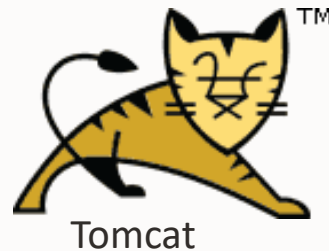
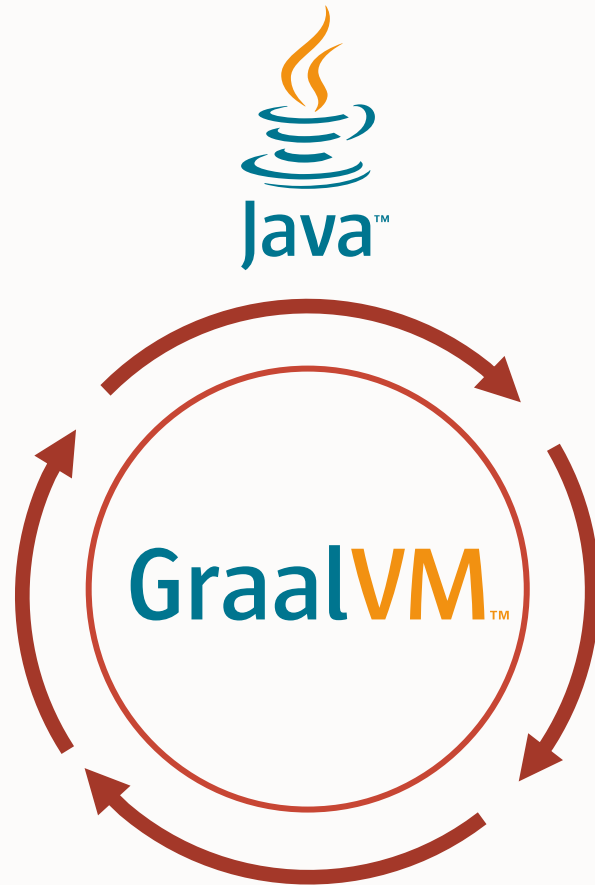




GraalVM Enterprise

Just-in-time compilation

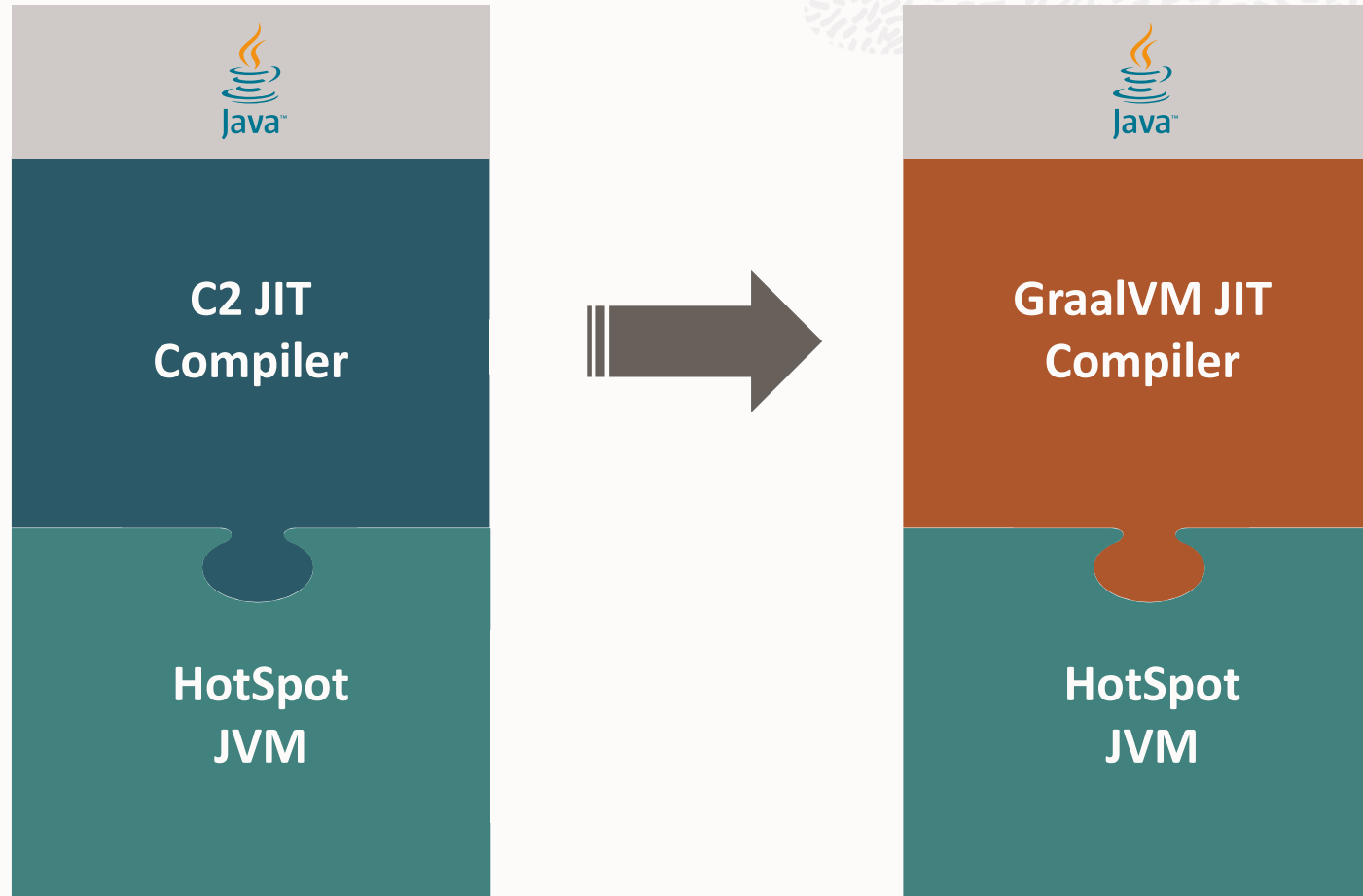
GraalVM Enterprise JIT Compiler—Ideal for traditional Java workloads



elasticsearch



GraalVM Enterprise: Oracle JDK + GraalVM JIT Compiler



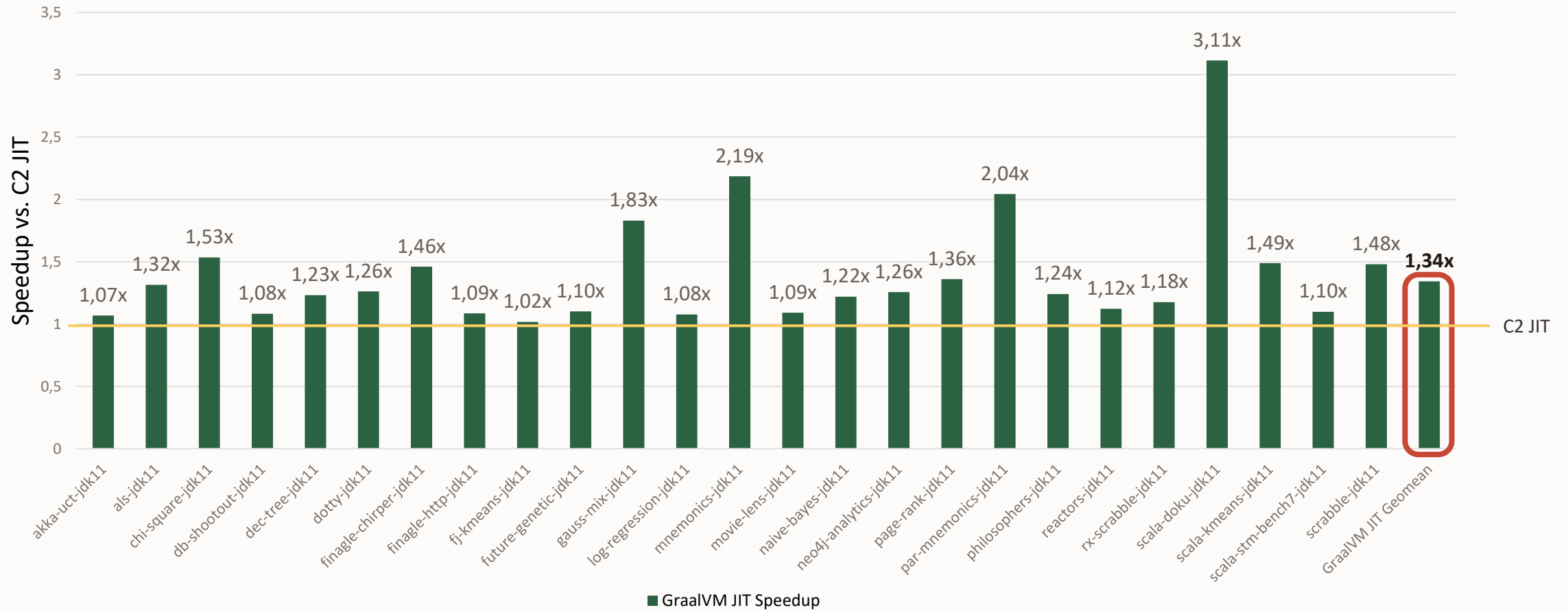
GraalVM
Enterprise



GraalVM Enterprise—Faster

Increased performance in real-world application benchmarks

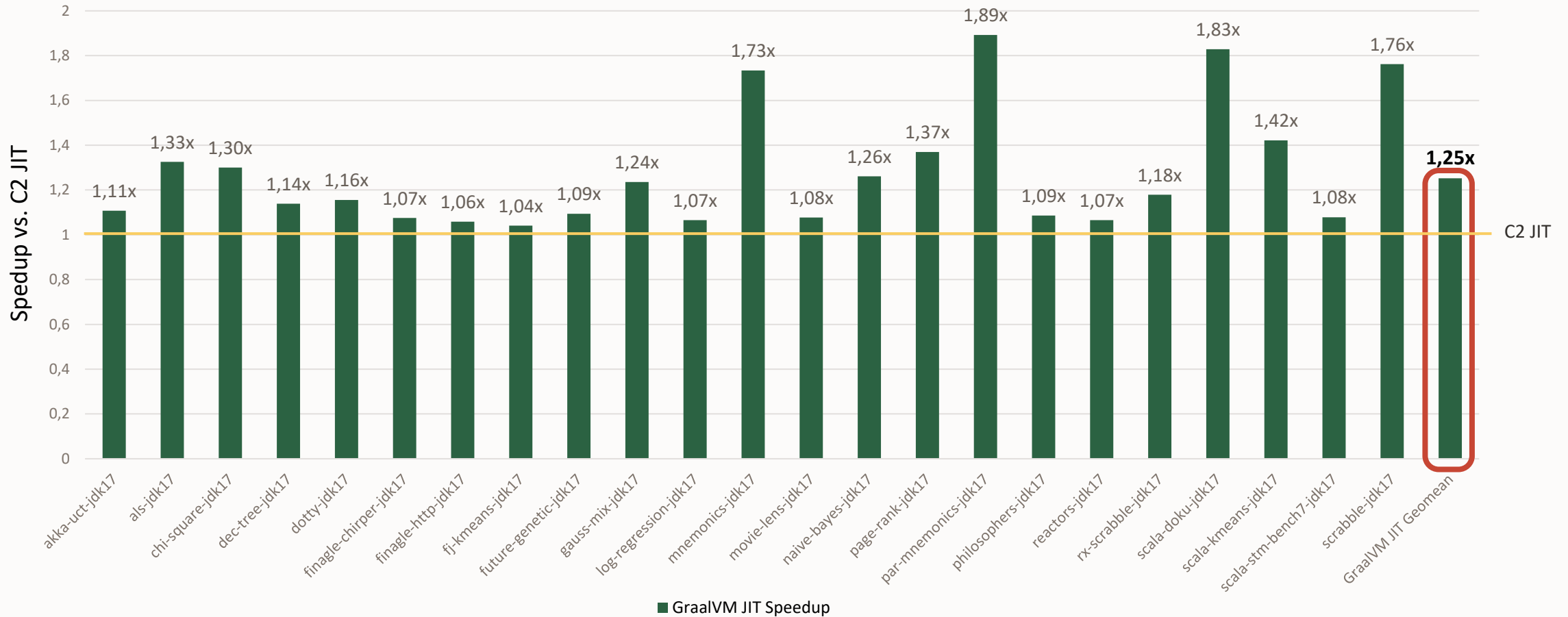
GraalVM 22.0 JDK 11 JIT vs. C2 JIT



GraalVM Enterprise—Faster

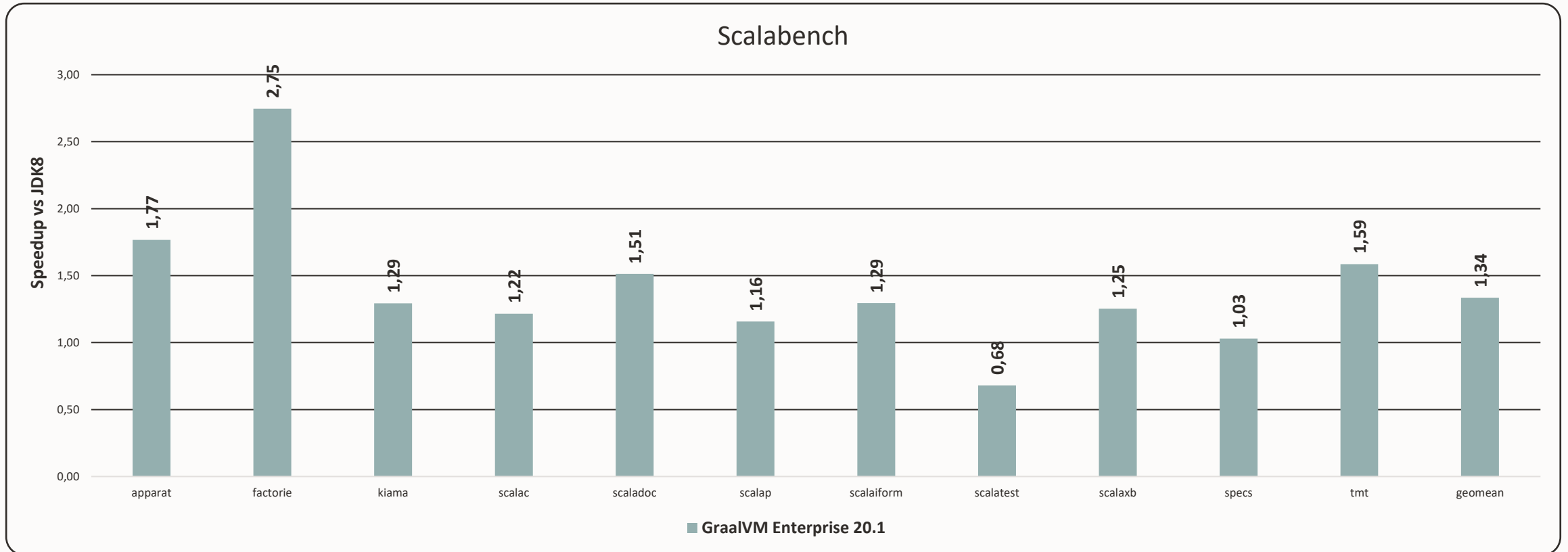
Increased performance in real-world application benchmarks

GraalVM 22.0 JDK 17 JIT vs. C2 JIT



GraalVM Enterprise—Faster Scala

Scalabench Up to 2.75x faster on GraalVM Enterprise

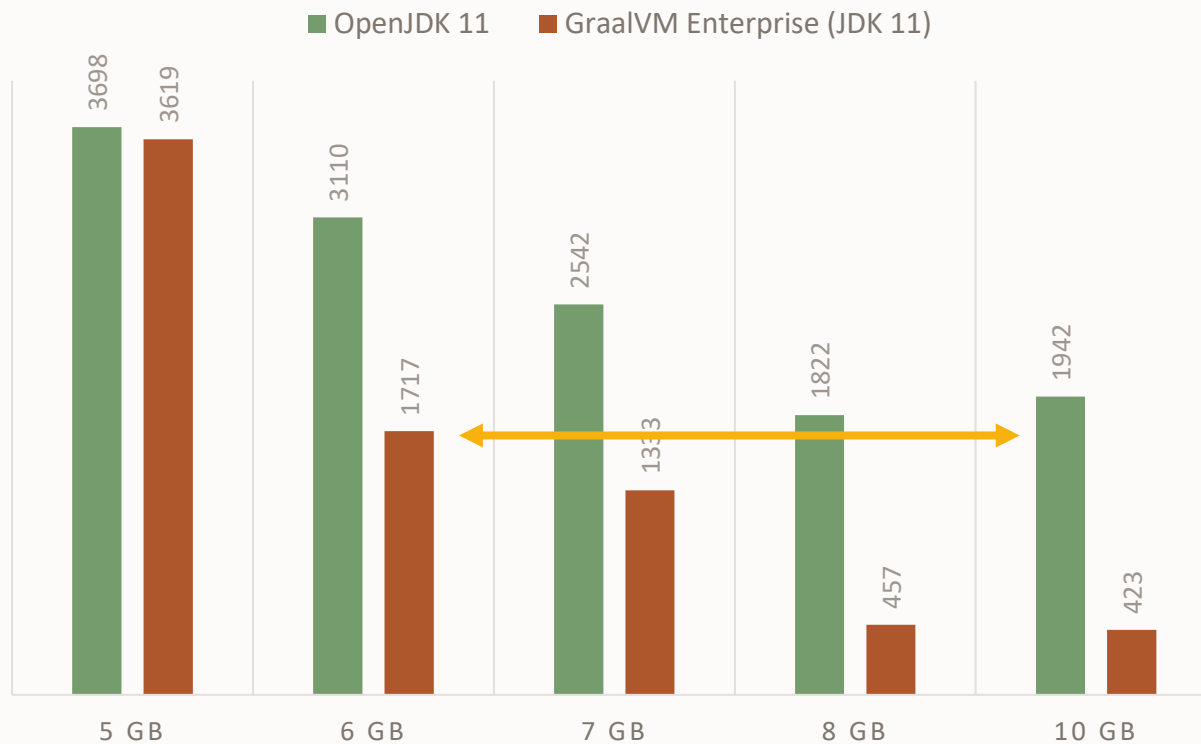


GraalVM Enterprise—**Leaner**

Higher performance with less memory



RUNNING TIME VS MEMORY, NAIVE-BAYES, JDK 11
(LOWER IS BETTER)



On the Renaissance “naive-bayes” benchmark, GraalVM Enterprise outperforms OpenJDK 11—regardless of the amount of available RAM.

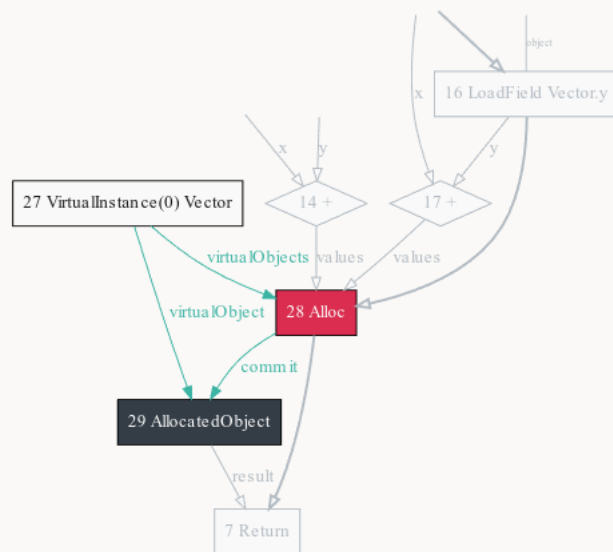
GraalVM Enterprise with 6GB outperforms OpenJDK with 10GB

Source: <https://blogs.oracle.com/graalvm/apache-spark%e2%80%94lightning-fast-on-graalvm-enterprise>



Seeing Escape Analysis Working

Chris Seaton, 16 December 2020



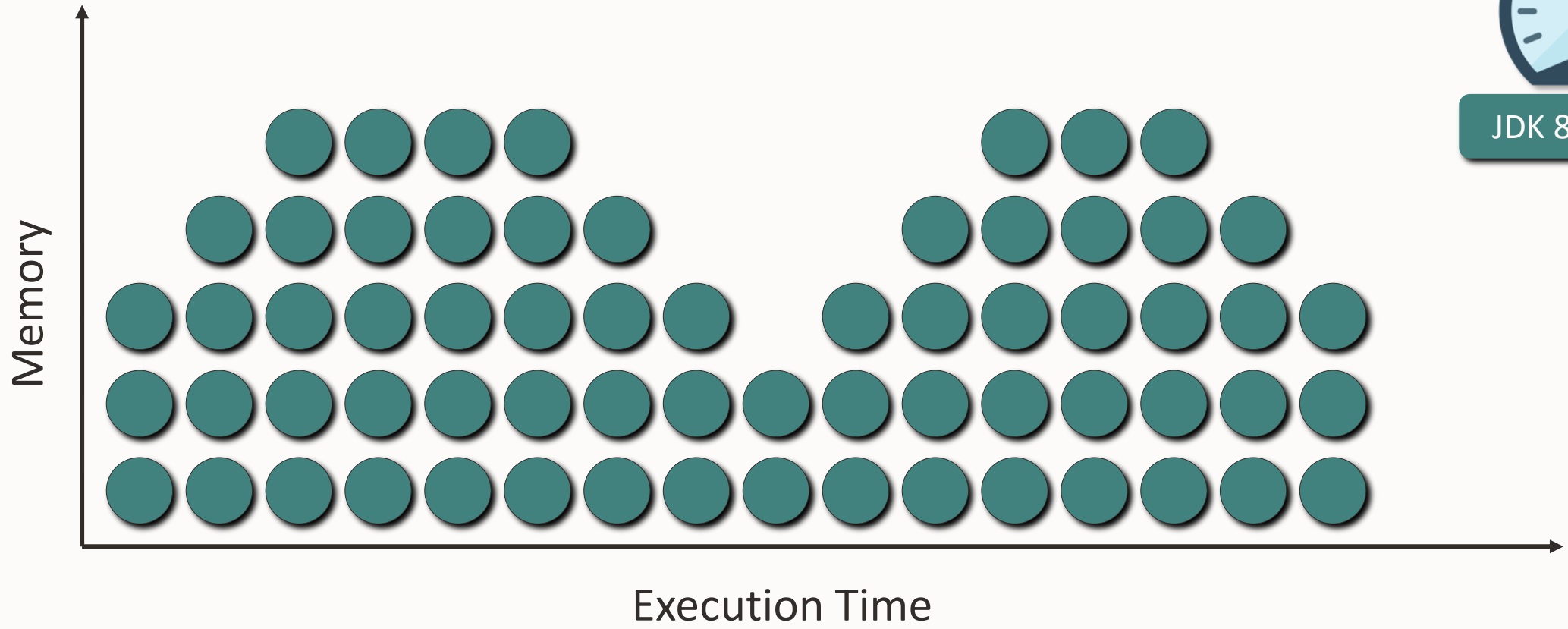
This article originally appeared in the [Java Advent 2020](#).

You may have heard of a compiler analysis phase called *escape analysis*. It informs an optimisation called *scalar replacement of aggregates* that removes unnecessary allocation of Java objects. I find that people often have a misunderstanding of what this optimisation really does and an under-appreciation of what it's capable of. We can know it better by seeing it working in practice.

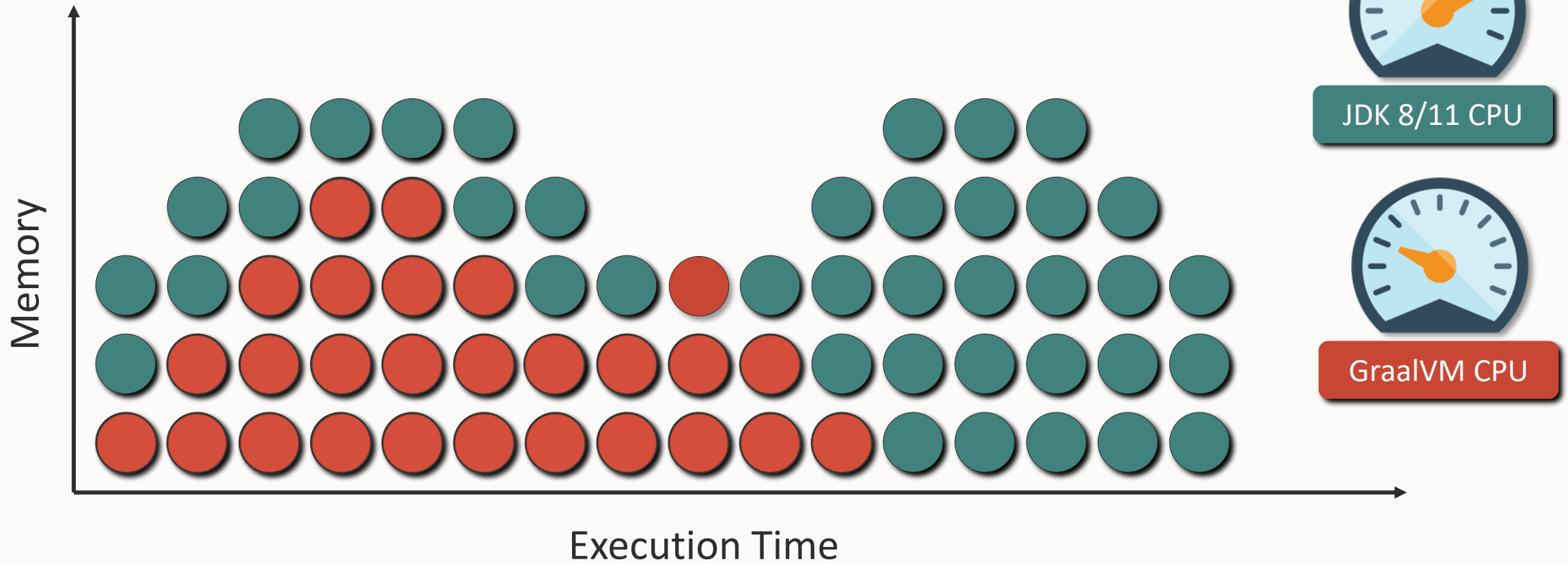
Why is GraalVM Enterprise **Faster**?



JDK 8/11 CPU



Why is GraalVM Enterprise **Faster**?





“Oracle GraalVM Enterprise Edition was the performance choice for our Dell EMC Servers. Java workload analysis and SPECjbb®2015 benchmark improving max-jOPS results by almost 8%.”

Kurtis Bowman

Director of Architecture, Server Office of the CTO



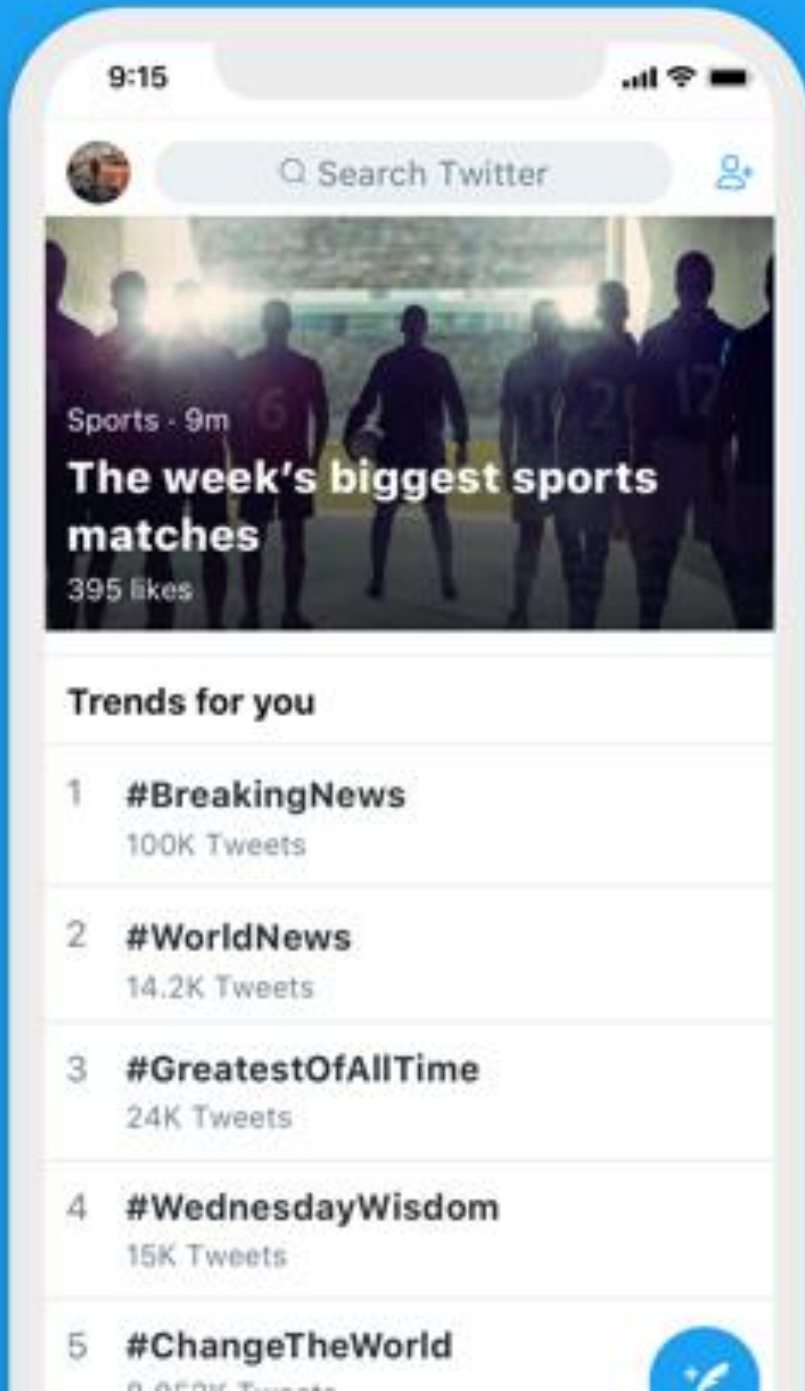


“We save a lot of money and CPU cycles”

10% performance increase
20% reduction in latency

Chris Thalinger

Staff System Engineer, Twitter



GraalVM Enterprise in Oracle Cloud Infrastructure

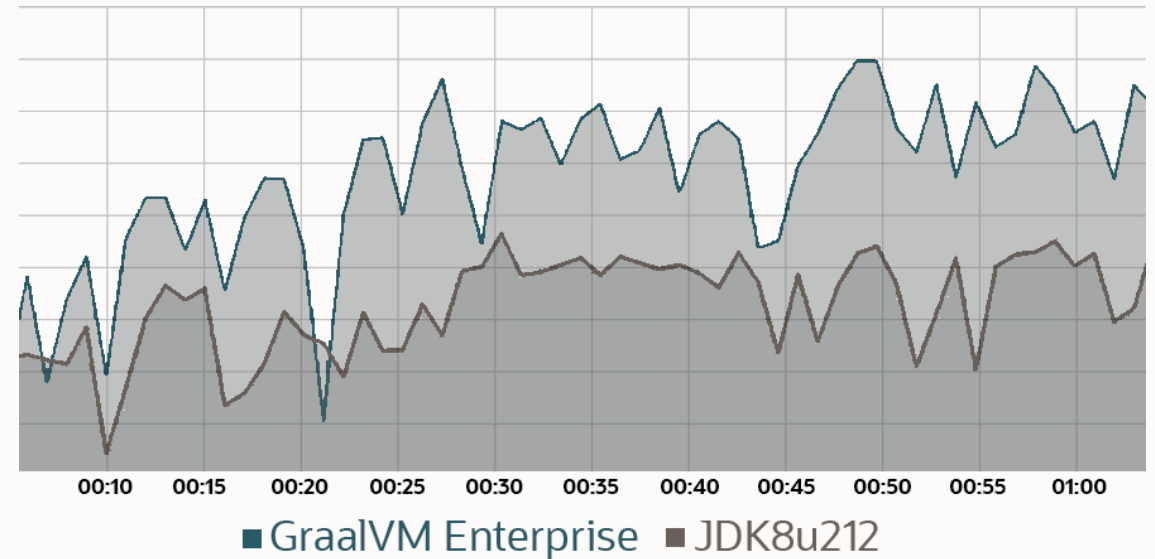
Real-world example



↓ **25% reduction**
in garbage collection time

↑ **10% improvement**
in transactions/sec

0 **Issues**
30+ Million core hours



[GraalVM Powers Oracle Cloud Infrastructure](#)



No code changes.



—

No code changes.

```
$ java -cp app.jar my.package.Main
```




GraalVM Enterprise

Ahead-of-time compilation

GraalVM

Facilitating the Spread of Knowledge and Innovation in Professional Software Development English Edition Write for InfoQ



Development

Architecture & Design

AI, ML

2,300,300 Jun unique visitors

- AUG 23** **InfoQ Live August**
Learn how cloud architectures help organizations take care of application and cloud security, observability, availability and elasticity. Register Now.
- OCT 24-28** **QCon San Francisco**
Understand the emerging software trends. Attend in-person on Oct 24-28, 2022.

InfoQ Homepage > Articles > Go Native With Spring Boot And GraalVM

JAVA

Go Native with Spring Boot and GraalVM



MAY 19, 2022 • 19 MIN READ

Key Takeaways

by **Josh Long** [FOLLOW](#)
Spring Developer Advocate at Tanzu, a division of VMware
reviewed by **Karsten Silz** [FOLLOW](#)
Full-Stack Java Developer & Contractor

- Spring Boot 3 and Spring Framework 6, due in late 2022, will have built-in support for native Java.
- For Spring Framework 5.x and Spring Boot 2.x users, Spring Native is the way to go.
- Spring Native provides integrations for a vast ecosystem of libraries.
- But Spring Native also ships a component model that allows you to extend native compilation support for other libraries.
- GraalVM AOT compilation offers a lot of possibilities with some (negotiable) costs.

Write for InfoQ

Building a Native Executable - x +
quarkus.io/guides/building-native-image
Waveney-sit.co.uk Bookmarks OCI-Cert Unseen University Work Chromebook Code Life

QUARKUS ABOUT LEARN C

[Back to Guides](#)

BUILDING A NATIVE EXECUTABLE

This guide covers:

- Compiling the application to a native executable
- Packaging the native executable in a container
- Debugging native executable

This guide takes as input the application developed in the [Getting Started Guide](#).

GraalVM

Building a native executable requires using a distribution of GraalVM. There are three distributions: Oracle GraalVM Community Edition (CE), Oracle GraalVM Enterprise Edition (E)

QCon Plus (Nov 29 - Dec 9); Ma

Docs Community Videos Blog Star 17,248

GraalVM™

High Performance. Cloud Native. Polyglot.

Java JavaScript Python Ruby R WASM

Download Get Started

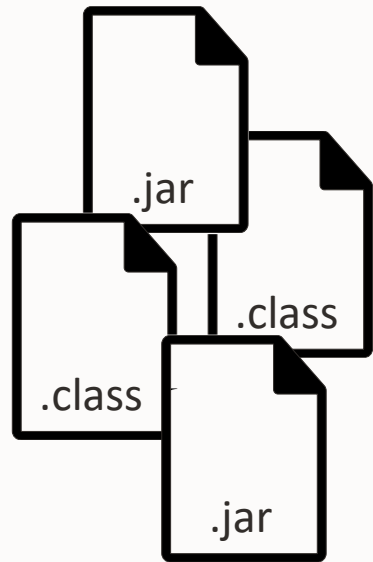
Conference Talk **Revolutionizing Java-Based Cloud Deployments with GraalVM**
MAY 16, 2022 youtube.com Watch

Oracle Developer Live: Java Innovations **GraalVM Native Image for Containerised Applications**
MAR 22, 2022 youtube.com Watch



GraalVM Enterprise Native Image—Ahead-of-time compiler & runtime

Microservices and Containers



Up to 5x less memory
100x faster startup



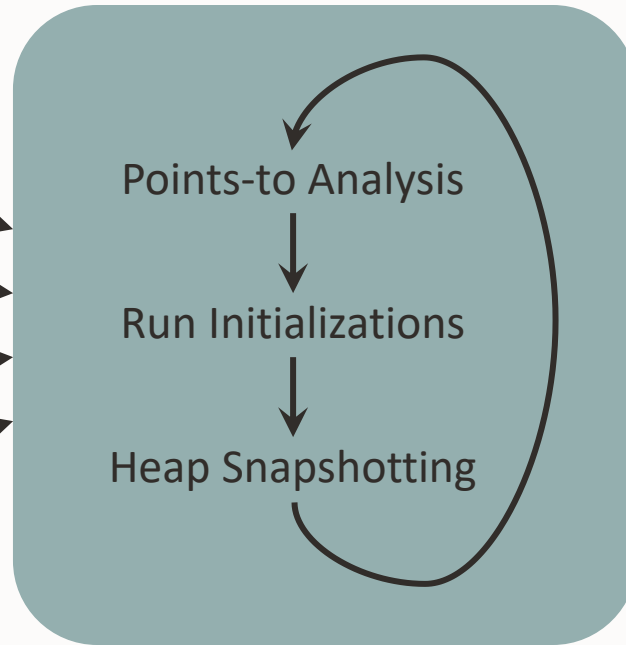
No code changes.

\$ native-image -cp app.jar my.package.Main

\$./main

Input:
All classes from application,
libraries, and VM

- Application
- Libraries
- JDK
- Substrate VM



Iterative analysis until
fixed point is reached

Ahead-of-Time
Compilation

Image Heap
Writing



Output:
Native executable



Closed World Assumption



- `native-image` assumes it knows everything
- Aggressively eliminates dead code
- Points to (static) analysis needs to see all bytecode
 - Eagerly loads referenced classes
- No loading of new classes at runtime
- Dynamic parts of Java require build time “[reachability metadata](#)” (configuration)
 - reflection, proxies, resources, JNI
 - Tooling for this (finding and telling)
 - [Github repo](#) for libraries



Reachability metadata



```
[
  {
    "condition": {
      "typeReachable": "com.zaxxer.hikari.util.ConcurrentBag"
    },
    "name": "[Lcom.zaxxer.hikari.util.ConcurrentBag$IConcurrentBagEntry;"
  },
  {
    "condition": {
      "typeReachable": "com.zaxxer.hikari.pool.PoolEntry"
    },
    "name": "[Ljava.sql.Statement;"
  },
  ...
]
```



GraalVM Enterprise Native Image

Supported by microservice frameworks and platforms



Ideal for Cloud Native



**CLOUD NATIVE
COMPUTING FOUNDATION**



Go Native with Spring Boot and GraalVM



LIKE



4



MAY 19, 2022 • 19 MIN READ

by



Josh Long

[FOLLOW](#)

Spring Developer Advocate at Tanzu,
a division of VMware

reviewed by



Karsten Silz

[FOLLOW](#)

Full-Stack Java Developer &
Contractor

[Write for InfoQ](#)

Key Takeaways

- Spring Boot 3 and Spring Framework 6, due in late 2022, will have built-in support for native Java.
- For Spring Framework 5.x and Spring Boot 2.x users, Spring Native is the way to go.
- Spring Native provides integrations for a vast ecosystem of libraries.
- But Spring Native also ships a component model that allows you to extend native compilation support for other libraries.
- GraalVM AOT compilation offers a lot of possibilities with some (negotiable) costs.

Not a silver bullet!



- Closed – world assumption
 - Tracing / manual configuration may be required
 - Reflection, dynamic class loading
- Retrofitting to existing code bases can require some work
- Compilation is not a security / obfuscation tool!
 - But there are security benefits (think log4j)
- Better for new development
 - Serverless
 - Microservices



Native Image and Cloud Native

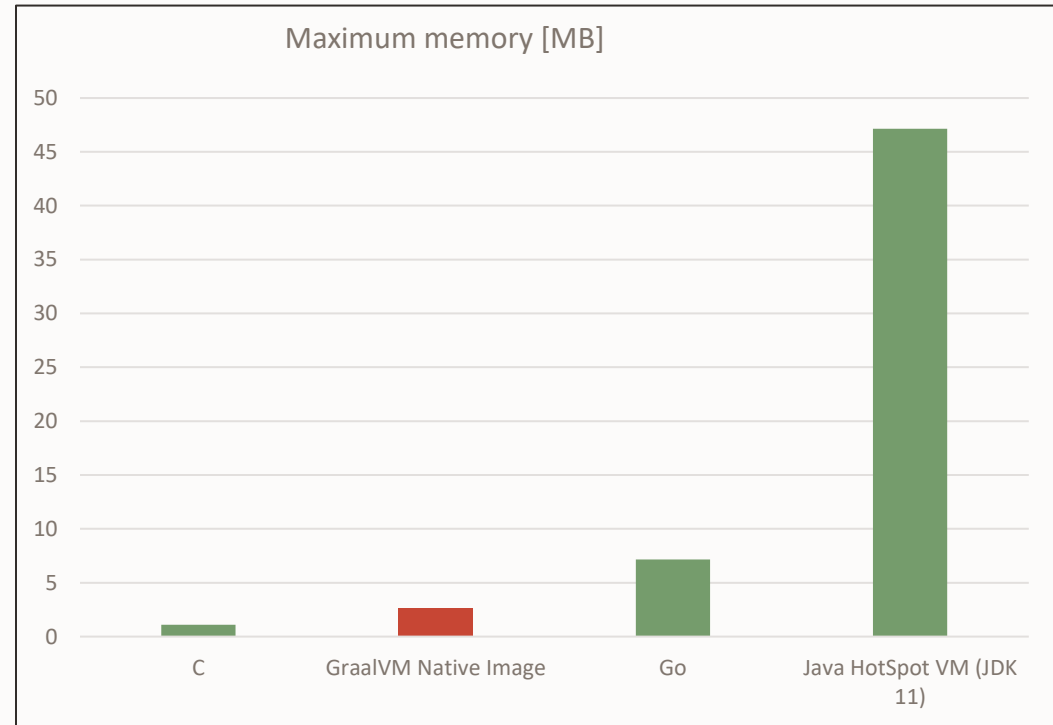


- Smaller runtime
 - Big jar → small binary
- Smaller containers
 - “distroless” → smaller attack surface
- Faster startup time
 - Initialisation during AOT compilation
 - Lower latency
- Lower memory usage



GraalVM Enterprise Native Image—Java productivity with C-like performance

Microservices and Containers

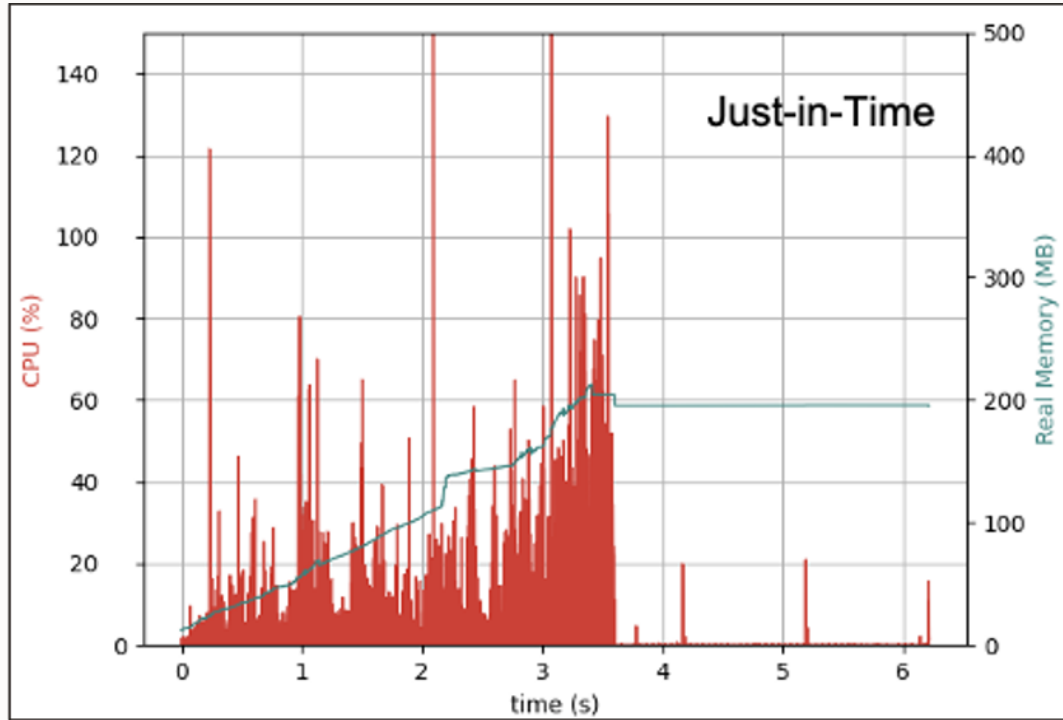


Lower compute requirements and faster execution reduces infrastructure/cloud costs

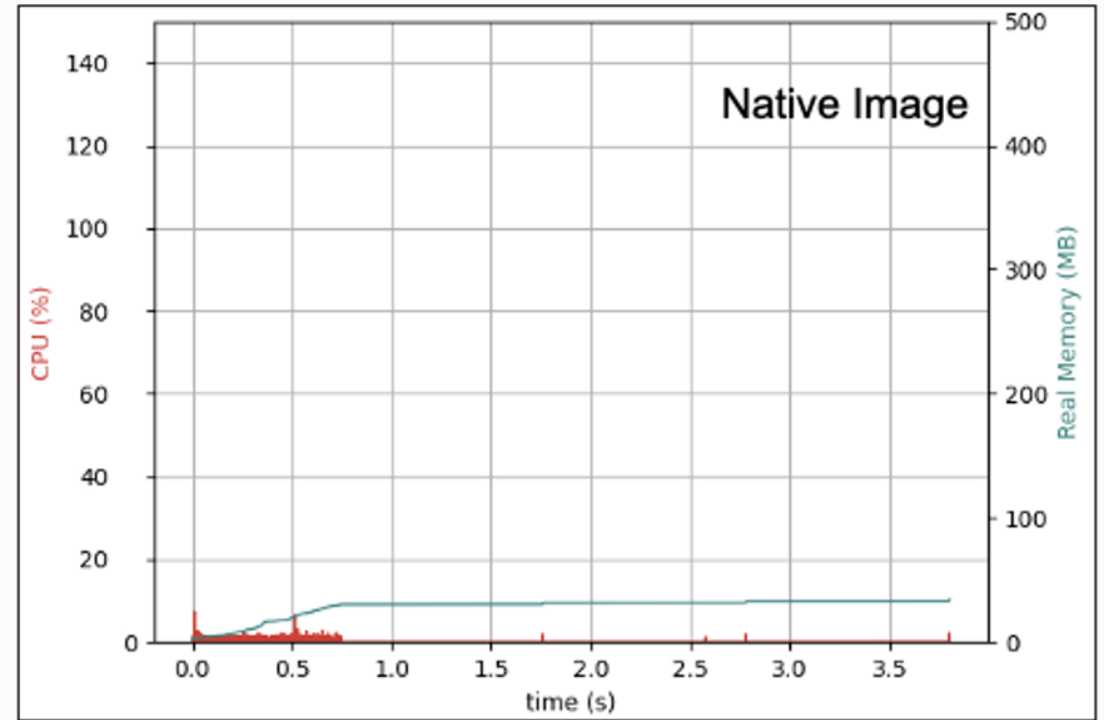


GraalVM Enterprise Native Image—drastic resource reductions

Microservices and Containers



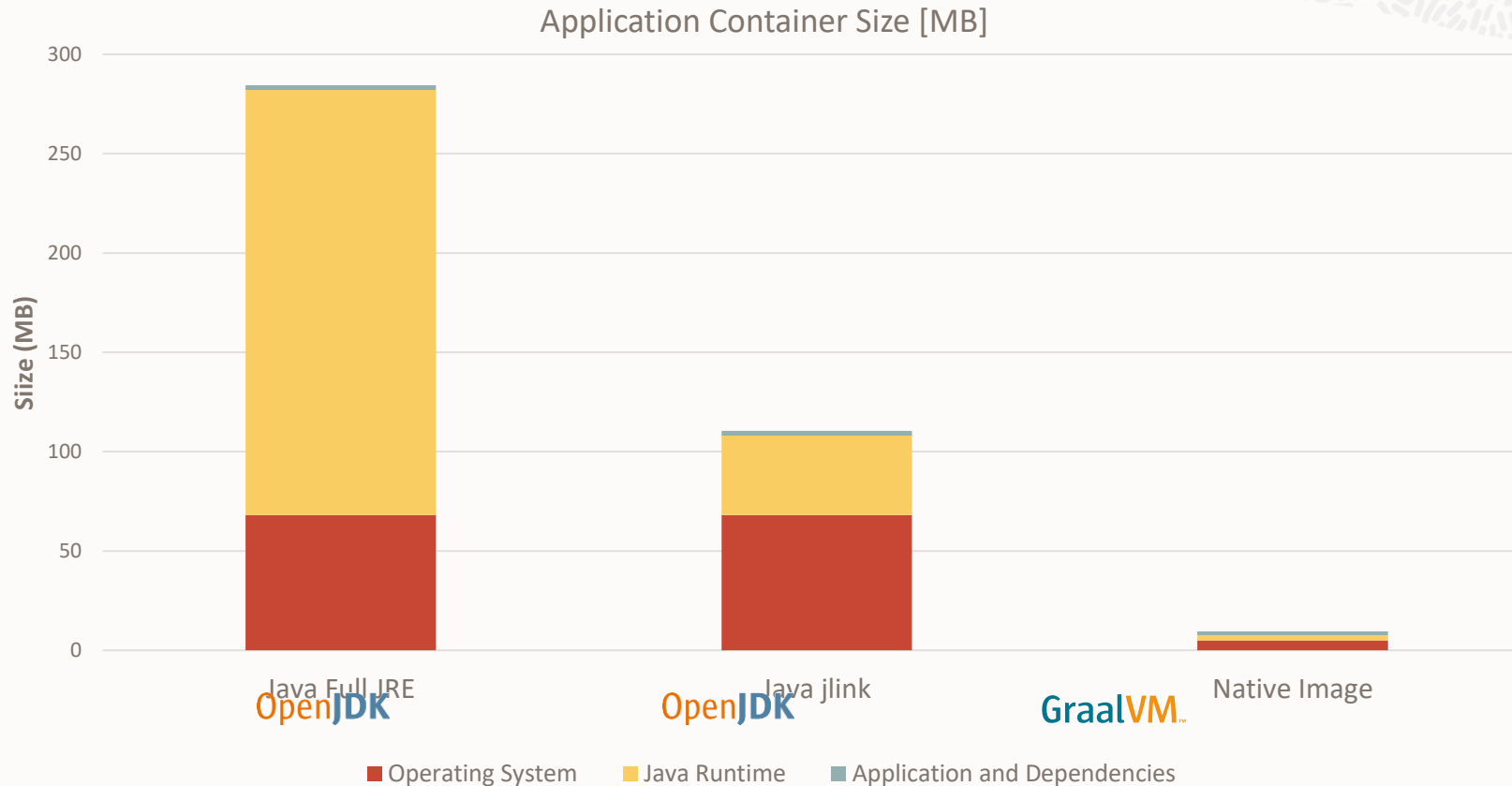
Microservice on GraalVM Enterprise (JIT mode)



GraalVM Enterprise Native Image generated native executable



GraalVM Native Image—Best Solution for Containerized Java Microservices and Containers



Lower compute requirements and faster execution reduces infrastructure/cloud costs



Example—Spring Boot with GraalVM Native Image

How fast is your PetClinic?

Sample	On the JDK	native-executable
petclinic-jdbc	Build: 9s Memory(RSS): 417M Startup time: 2.6s	Build: 194s +2050% Memory(RSS): 101M -75% Startup time: 0.158s -94%



Example—Spring Boot with GraalVM Native Image

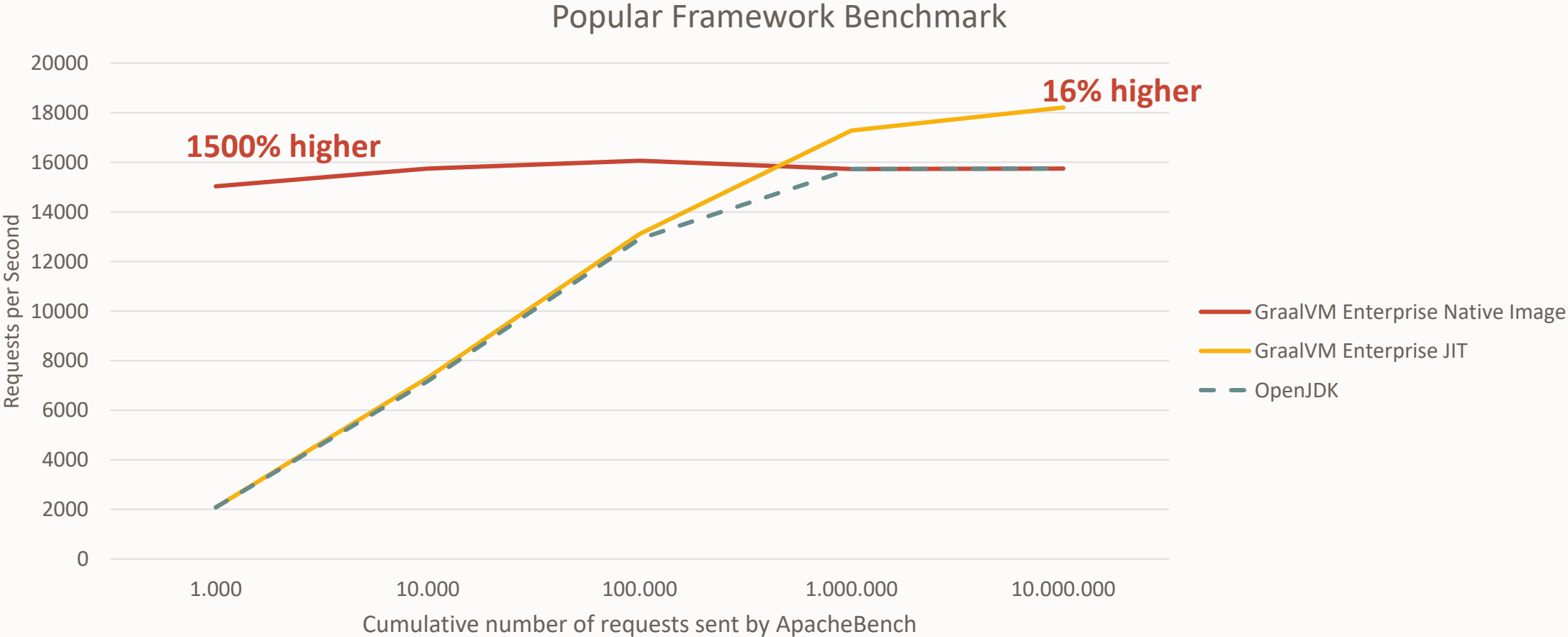


On the JDK	native-executable
Build: 9s	Build: 194s
Memory(RSS): 417M	Memory(RSS): 101M
Startup time: 2.6s	Startup time: 0.158s

+2050%
-75%
-94%



GraalVM Enterprise throughput



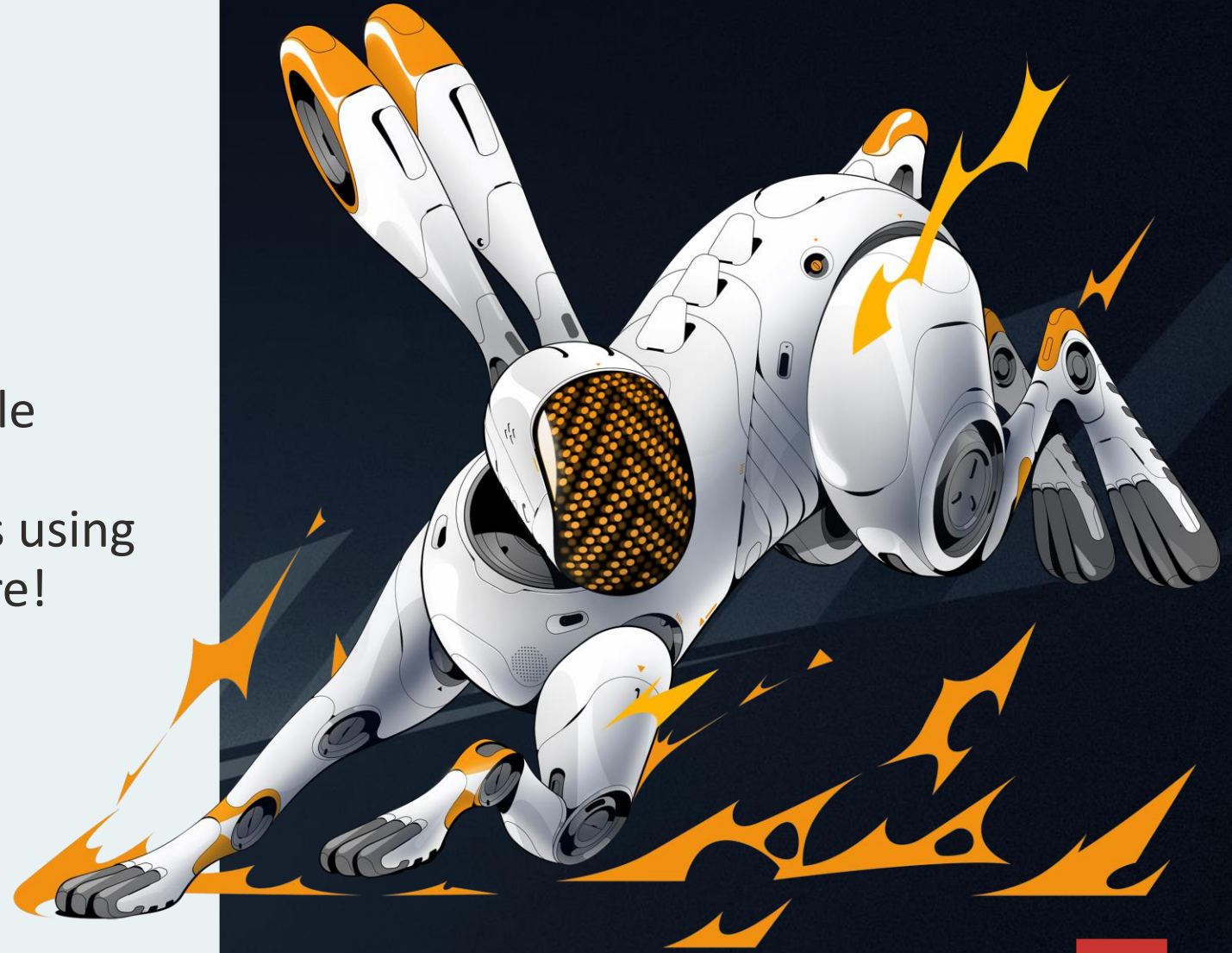
Scaffolding

—
Graal Cloud Native

GraalVM™

Graal Cloud Native

Making it easy to build cloud-portable applications that leverage powerful platform provided managed services using *Spring*, *Micronaut*, *Helidon*, and more!



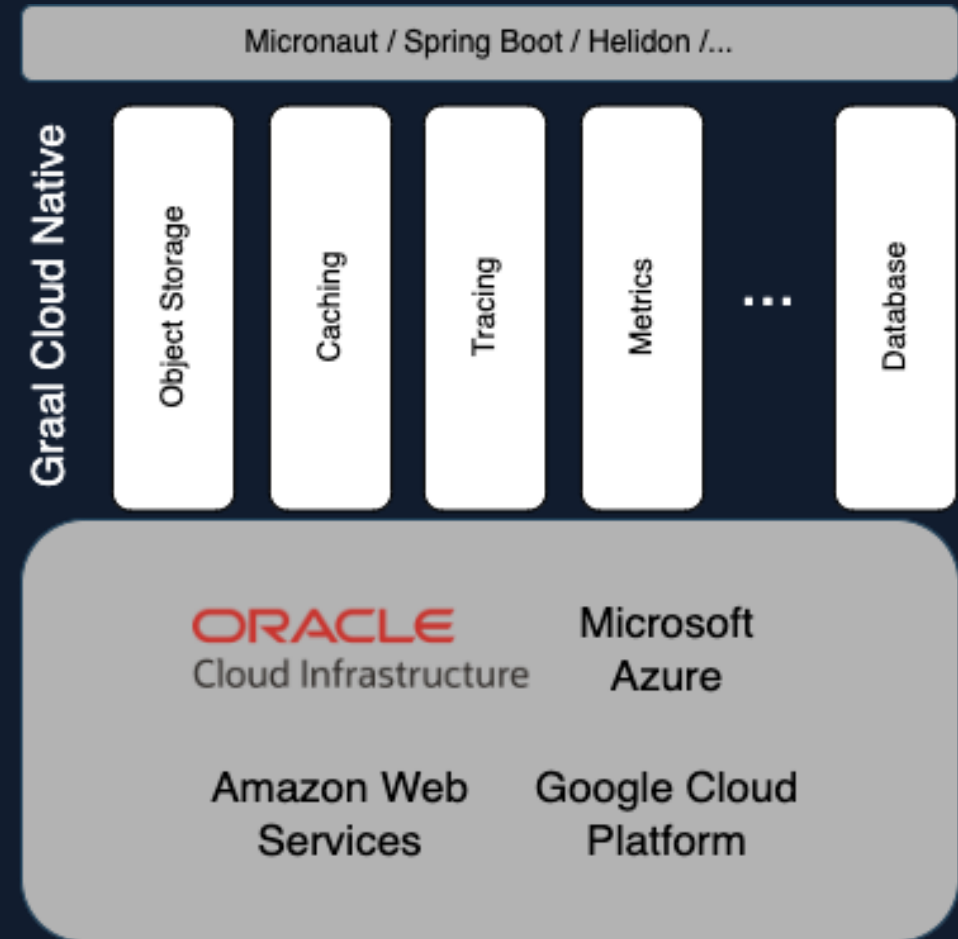
Graal Cloud Native



Graal Cloud Native lets you take full advantage of powerful cloud services without coupling to proprietary platform APIs.

Graal Cloud Native solves the problem of the lack of standard cross-cloud APIs and makes application portability possible.

Use Graal Cloud Native to leverage services like object-storage, monitoring, authentication, secret management, and deploy to popular cloud platforms.



Graal Cloud Native

VS Code Tools for OCI



- Powerful VS Code extensions that make it easy to onboard Java developers onto OCI
- Full support for DevOps projects including Git source control, build pipelines, container repositories, etc. from within VS Code
- OCI Application Dependency Manager integration to detect critical vulnerabilities during application development
- Compatible with OCI Cloud Editor and Cloud IDE

```
! build_spec.yaml
! build_spec.yaml
1 version: 0.1
2 component: build
3 timeoutInSeconds: 20000
4 runAs: root
5 shell: bash
6 env:
7   variables:
8     "JAVA_HOME" : "/usr/lib64/graalvm/graalvm22-ee-java17"
9 steps:
10  - type: Command
11    name: "Pre-yum repo checks"
12    command: |
13      printf "\n\n Contents of graal_spec.yaml file:\n $(cat graal_spec.yaml) \n\n"
14      printf "Contents of /etc/yum.repos.d:\n $(ls -alh /etc/yum.repos.d/) \n"
15      printf "Contents of /etc/pki/rpm-gpg:\n $(ls -alh /etc/pki/rpm-gpg/) \n"
16  - type: Command
17    name: "Temporary workaround: Set up the yum repo"
18    command: |
19      (echo -e '[ol7_oci_included]' && \
20      echo 'name=Oracle Software for OCI users on Oracle Linux $releasever ($basearch)' && \
21      echo 'baseurl=https://yum-phx.oracle.com/repo/OracleLinux/OL7/oci/included/$basearch/' && \
22      echo 'gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle' && \
23      echo 'gpgcheck=1' && \
24      echo 'enabled=1') > /etc/yum.repos.d/oci-included-ol7.repo
25      yum-config-manager --enable ol7_optional_latest
26  - type: Command
27    name: "Post-yum repo checks"
28    command: |
29      printf "Contents of /etc/yum.repos.d:\n $(ls -alh /etc/yum.repos.d/) \n"
30  - type: Command
31    name: "Install GraalVM Enterprise 22.1 Java 17 - JDK and Native Image"
32    command: |
33      yum -y install graalvm22-ee-17-native-image
34  - type: Command
35    name: "Set the PATH here. JAVA_HOME already set in env > variables above."
```



Graal Cloud Native Benefits

Developer Productivity

Simplified service usage eliminates the need to learn proprietary platform APIs.

Advanced compile-time validation to reduce dev/test/debug cycle.

Easily Leverage Cloud Platform Services

Inject cloud services right into an application exactly where they are needed.

All major services supported out of the box.

Apps start fast and use fewer resources

Out-of-the-box compatibility with GraalVM Native Image let's applications start up to 100x faster than when running on the JVM, deliver immediate peak performance, and require less memory and CPU.

Multicloud by design

Deploy to different cloud providers with no code changes.

Support for all the major cloud platforms.

Graal Cloud Native with GraalVM

developer.oracle.com/java/graalvm/cloud-native/

Waveney-slt.co.uk | Bookmarks | OCI-Cert | Unseen University | Work | Chromebook | Code | Life

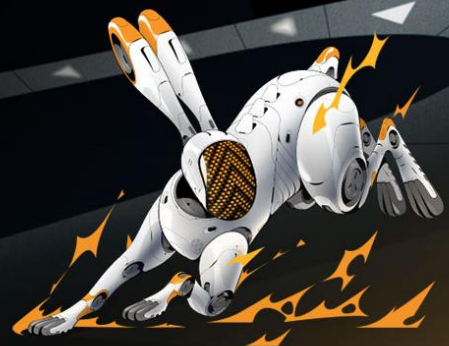
ORACLE | Products and Technologies | Learn | Community | Downloads | View Accounts | Get Started

Developer Resource Center / Graal Cloud Native


Graal Cloud Native

Graal Cloud Native (GCN) is a full stack framework for building portable cloud-native applications using the Java ecosystem on top of managed services in the public cloud. GCN is built on top of the GraalVM native image, the Micronaut® framework, public cloud SDKs, popular drivers, and a set of open source libraries tested, certified, and supported by Oracle. Oracle has optimized this stack for native image as well as for cloud portability. With GCN, cloud users can focus less on language portability and more on portability across cloud vendors.

[Try Oracle Cloud Free Tier](#)




Benefits of Graal Cloud Native



Developer Productivity

Save time with out-of-the-box integrations and compile-time validation with common developer technologies.


[Building Apps with Micronaut® \(20:37\)](#)
[Code on GitHub](#)



Easily Leverage Oracle Cloud Services

Micronaut® can inject Oracle Cloud services right into an application exactly where they are needed.


[Leverage Oracle Cloud Services \(3:53\)](#)
[Code on GitHub](#)



Start Fast with Fewer Required Resources

Graal Cloud Native applications compiled by GraalVM Native Image start up to 100x faster than when running on the JVM, operate at peak performance immediately, and require less memory and CPU.

[Build Performant Apps \(3:03\)](#)
[Code on GitHub](#)



Multicloud by Design

Deploy to different cloud providers with no code changes.

[Multi-Cloud by Design \(13:13\)](#)
[Code on GitHub](#)

Get Started with Graal Cloud Native



Resources



- Home page <https://www.graalvm.org/>
- Twitter <https://twitter.com/GraalVM>
- Medium <https://medium.com/graalvm>

- Luna Labs
 - <https://luna.oracle.com/>
 - Use Oracle login (not OCI), search for GraalVM
- Live Labs
 - <http://bit.ly/golivelabs>
- “GraalVM for Dummies” ebook
 - <https://go.oracle.com/LP=105746>



Summary



- With cloud, you pay for what you use
- You need to be as efficient as possible
- Run your Java applications more efficiently with GraalVM EE
- GraalVM EE is included with OCI
- The OCI price – performance advantage is even bigger for Java apps
- Try GraalVM EE & OCI today



Let's head to the lab...

Thank you



Any Questions?



ORACLE