Brought to you by:

ORACLE

GraalVM

for dummies®

A Wiley Brand

Accelerate application performance

Optimize CPU and memory usage

Reduce time to market for new apps

Lawrence Miller

Oracle Special Edition

About Oracle

Oracle Cloud is the industry's broadest and most integrated public cloud, offering a complete range of services across SaaS, PaaS, and IaaS. It supports new and existing cloud environments and development runtimes like Oracle GraalVM Enterprise to accelerate application performance, lower infrastructure costs, and deliver the best solution for deploying microservices, on premises and in the cloud.

For more information, please visit us at **oracle.com/java** and **oracle.com/graalvm.**

GraalVM

GraalVM

Oracle Special Edition

by Lawrence Miller

for dummies®
A Wiley Brand

GraalVM For Dummies®, Oracle Special Edition

Published by

John Wiley & Sons, Inc.

111 River St.

Hoboken, NJ 07030-5774

www.wiley.com

ISBN 978-1-119-76642-1 (pbk); ISBN 978-1-119-76646-9 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Development Editor: Elizabeth Kuball

Copy Editor: Elizabeth Kuball

Acquisitions Editor: Ashley Coffey

Editorial Manager: Rev Mengle

Business Development Representative: William Hull

Production Editor: Mohammed Zafar Ali

Table of Contents

Introduction

Businesses everywhere are under constant pressure to be smarter, faster, and more agile. They must deliver innovative new software services for their customers and automate internal business processes. This requires software to scale and often causes IT costs to grow. Thus, applications must run faster and consume fewer resources.

As organizations move applications to the cloud, they must look for ways to reduce costs and improve efficiency. Application development for cloud environments takes advantage of new architectures like microservices in containerized environments.

In this book, you see how the technology addresses the challenges of modern application development:

- Built on trusted Oracle Java Standard Edition (SE), adding a state-of-the-art just-in-time (JIT) compiler that accelerates application performance, lowers required memory per operation, and reduces operating costs.
- Supports multi-language applications running on a single Java Virtual Machine (JVM) and enables different programming language libraries to efficiently interoperate.

- Leverages more than a decade of research and development, including technology to compile Java applications to native binaries, which is the ideal platform for cloud-native application development.
- Provides customers with 24/7 access to the experienced Oracle GraalVM Enterprise Edition support team.

About This Book

GraalVM For Dummies consists of six chapters that explore the following:

- The business need for GraalVM and the basics of GraalVM and GraalVM Enterprise (Chapter 1)
- How GraalVM accelerates application performance (Chapter 2)
- Using GraalVM to develop cloud-native apps (Chapter 3)
- How multi-language support in GraalVM increases productivity (Chapter 4)
- Different GraalVM use cases (Chapter 5)
- Key capabilities and benefits of GraalVM Enterprise (Chapter 6)

Foolish Assumptions

It's been said that most assumptions have outlived their uselessness, but I assume a few things nonetheless!

Mainly, I assume you are a chief information officer (CIO), a chief technology officer (CTO), a vice president or director of development, or an application developer. As such, this book is written primarily for technical readers.

Icons Used in This Book

Throughout this book, I occasionally use icons to call out important information. Here's what to expect.

This icon points out information you should commit to your nonvolatile memory, your gray matter, or your noggin!

If you seek to attain the seventh level of NERD-vana, perk up! This icon explains the jargon beneath the jargon!

Tips are appreciated, but never expected, and I sure hope you'll appreciate these useful nuggets of information.

These alerts point out the stuff your mother warned you about. Well, probably not, but they do offer practical advice to help you avoid potentially costly or frustrating mistakes.

Beyond the Book

If you find yourself at the end of this book thinking, "Gosh, this was an amazing book — where can I learn more?," just head to `https://oracle.com/graalvm`.

IN THIS CHAPTER

- Looking at modern business drivers
- Discovering GraalVM
- Getting more out of GraalVM with Oracle GraalVM Enterprise

Chapter 1
Unlocking GraalVM

In this chapter, you explore modern business challenges that are driving the need for faster software running in on-premises data centers and in the cloud. You also see how Oracle GraalVM Enterprise further enhances the business value you're getting out of your deployed applications.

Recognizing Business Needs and Motivations

Businesses must constantly find new ways to attract and retain customers while also increasing agility,

maximizing productivity, and reducing costs. The key digital transformation initiatives for enterprises are modernizing applications at the edge.

Businesses that operate their own data centers are looking for ways to increase application performance. To avoid adding costly compute resources, they're looking for application efficiencies to get more out of their existing infrastructure investments. They're also moving workloads to the cloud for savings from a "pay-as-you-go" model. Running applications faster and more efficiently would reduce cloud resource consumption costs.

At the same time, companies are changing the way in which they build and deliver software solutions. They're migrating away from the *n*-tier enterprise platforms toward horizontally scalable container-based architectures and microservices.

Regardless of whether you're running traditional enterprise applications on premises or modern cloud-native applications, businesses can improve efficiency by enabling applications to run faster and consume fewer server CPU and memory resources.

What Is GraalVM Enterprise?

GraalVM Enterprise is a high-performance runtime built on Oracle Java SE. It adds new compiler optimizations that deliver the best performance for running Java

applications and microservices on premises or in the cloud. It provides a just-in-time (JIT) compiler that accelerates Java applications without code changes. GraalVM enhances the Java ecosystem by offering a compatible and better-performing Java Development Kit (JDK).

GraalVM includes a Native Image utility to compile applications ahead-of-time into native executables. The executables are smaller, start nearly instantaneously, and consume a fraction of the compute resources needed to run the same Java application on a Java Virtual Machine (JVM) (see Figure 1-1). It makes GraalVM ideal for cloud deployments and microservices. The native executables are comparable to native C/C++ programs in startup, memory usage, and performance, without leaving the powerful ecosystem of Java.

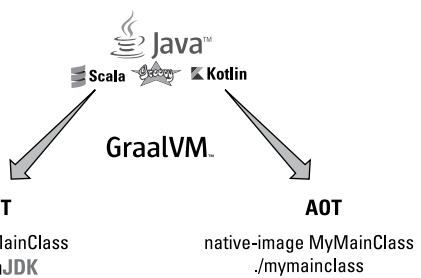

FIGURE 1-1: GraalVM JIT and AOT compiler optimizations.

GraalVM supports different programming languages including Java and JVM languages like Scala, Kotlin, and so on, as well as other languages such as JavaScript, Node.js apps, Ruby, Python, and others. The GraalVM Language Implementation Framework, in conjunction with the sophisticated GraalVM JIT compiler (Truffle), enables programs written in multiple languages to:

- Run on the JVM.
- Interoperate with shared memory and shared data structures.
- Be compiled down to native machine code for excellent performance.

GraalVM is the fastest available Java runtime. So, if you're interested in improving the performance of your individual Java services or you want to reduce your hardware or compute costs, you should look at GraalVM Enterprise.

GraalVM Enterprise follows the open-core model with these added enterprise features:

- Additional patented compiler optimizations for both JIT and AOT compiled code (native image) for significantly better performance
- Enhance garbage collection in native executables including support for G1 GC

- Lower memory and CPU consumption as a result of superior object allocation elimination and compressed pointers
- Resource limits for less trusted code (for example, native libraries written in C and C++)
- 24/7/365 support from Oracle backed by the expertise of the GraalVM team

REMEMBER

Throughout this book, I use *GraalVM* to refer to Oracle GraalVM Enterprise and *GraalVM Community* to refer to the open-source version of GraalVM.

Exploring GraalVM Enterprise

Built on trusted and secure Oracle Java SE, GraalVM incorporates more than a decade of research and development innovations into advanced optimizing compiler technology. It's the ideal platform for application modernization.

GraalVM Enterprise is available with a Java SE subscription at no additional cost. In addition to the core technology, you get Oracle support with access to quarterly performance, scalability, and security updates.


TIP

Key business benefits of GraalVM include the following:

- Speeds up application performance by an average of up to 55 percent compared to the OpenJDK builds it is based on, without any code changes
- Allows building cloud-native, precompiled applications which have startup and memory usage characteristics of native C/C++ or Go.

IN THIS CHAPTER

- Leveraging innovative compiler optimization algorithms
- Getting more performance out of existing infrastructure
- Scaling applications to improve performance

Chapter 2

Supercharging Application Performance

GraalVM significantly improves the performance of all Java applications and of applications written in other Java Virtual Machine (JVM) languages, with no code changes. In this chapter, you find out how GraalVM accelerates application performance with innovative compiler optimizations for better performance on existing infrastructure and at cloud scale.

Looking at Compiler Optimizations

The JVM in any Java Development Kit (JDK) is a highly optimized runtime with years of engineering effort put into making it fast. GraalVM runs applications faster by leveraging the proven JVM infrastructure of the Oracle JDK and an advanced optimizing just-in-time (JIT) compiler that produces more efficient machine code. GraalVM relies on dozens of optimizations that work together, such as partial escape analysis, partial loop unrolling, aggressive priority and polymorphic inlining, and sophisticated data flow analysis techniques. For example, partial escape analysis benefits from inlining to determine when it's possible to avoid allocating objects, reducing both instruction cycles and the burden on the garbage collector.


TIP

Even the smallest applications, like a micro-benchmark for example, could require more than 100 optimizing transformations (called *phases*) in GraalVM. So, having access to the full collection of the best optimizations available is what makes GraalVM the fastest runtime for Java applications.

The Renaissance benchmark suite is a good test for real-world workloads using individual benchmarks from modern Java libraries and frameworks. This provides a better prediction of Java code.

Figure 2-1 shows the results for the Renaissance benchmarks for GraalVM, which shows an average increase of 55 percent over OpenJDK 8 with some benchmarks running over four times faster.

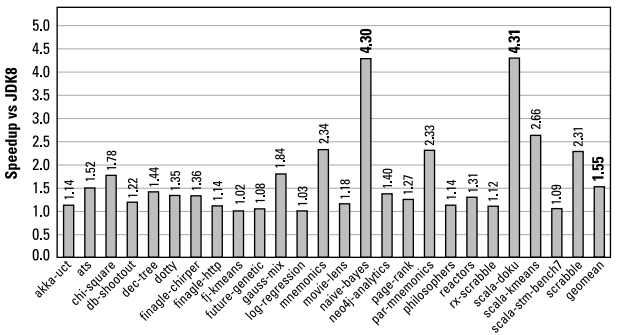

FIGURE 2-1: GraalVM Renaissance benchmark performance.

TIP

In general, almost all JVM applications will see a 10 percent to 20 percent speed increase using GraalVM compared to OpenJDK 8.

Even very large, complex, and already optimized existing JDK applications see performance improvements when executed on GraalVM. When the Oracle WebLogic Server team certified the application server on GraalVM Enterprise, they found performance improvements across all internal benchmarks — with no code changes.

The GraalVM compiler does even more for Scala application performance. The Scala language has more abstractions for the compiler to work with, so large Scala applications gain 30 percent to 40 percent better performance. Figure 2-2 shows that on the Scalabench test suite, GraalVM outperforms OpenJDK.

The GraalVM compiler includes more than 60 separate compiler optimization algorithms (or phases), of which 27 are patented (for example, new techniques for vectorizing complex programs, large-scale escape analysis, and code specialization).

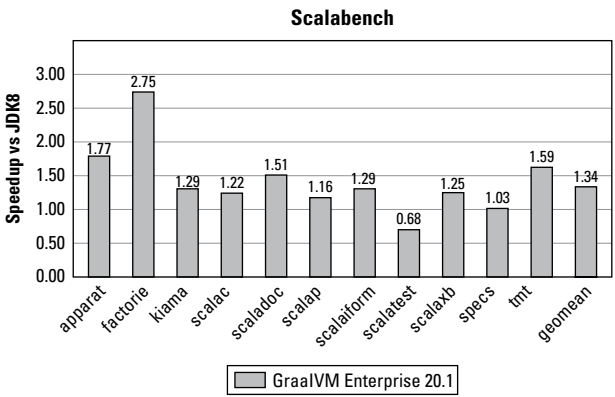

FIGURE 2-2: GraalVM Scalabench performance relative to JDK 8.

Enabling Faster Workloads with Existing Infrastructure

Faster application execution reduces the response time for user requests and frees up CPU and memory sooner. These resources can be used to handle other requests or by other applications running on the same server. GraalVM enables servers to handle more increasing workload requests with the same computing infrastructure, reducing the need to purchase additional hardware. Benchmarks of large Apache Spark clusters running thousands of nodes processing petabytes of data on the Renaissance Suite show that GraalVM is 1.6 times faster on average, potentially requiring 37 percent less hardware for the same computation.

REMEMBER

GraalVM's optimization of CPU and memory resource usage for applications can help companies reduce their need for future capital expenditures (CapEx) for on-premises infrastructure, as well as reducing cloud operating expenditures (OpEx).

Scaling Performance with Faster Cold Start

Faster application execution also helps applications scale in the cloud. Cloud-native applications are often built using microservices with lightweight frameworks.

Figure 2-3 shows the results of a performance evaluation of a small microservice fetching data from the database and rendering it to the user. The GraalVM-generated native executable runs at the peak rate almost immediately, while the JIT compiled JDK12 application needs to warm up to achieve the same peak performance. Native executables are ideal for short-lived microservices because they start incredibly fast, while traditional long-running Java workloads can tolerate the slower warmup performance profile of a traditional JIT compiler.

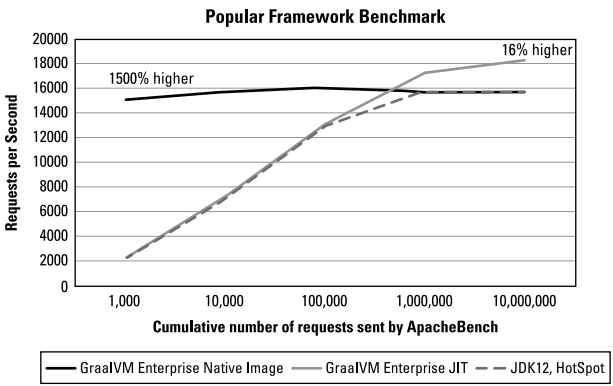

FIGURE 2-3: Throughput over time of GraalVM using JIT compilation and Native Image versus JDK 12.

IN THIS CHAPTER

- Developing microservices and cloud-native apps
- Ensuring high performance for production microservices
- Revisiting the cloud value proposition

Chapter 3

Exploring the Ideal Platform for Microservices and Cloud-Native Apps

In this chapter, you discover why GraalVM is the right application runtime for modern cloud-native apps built on a microservices architecture and how GraalVM helps enterprises reduce their cloud costs with faster application performance and a lower memory footprint.

Microservices Development Environment

Unlike traditional monolithic applications (which are self-contained) and *n*-tier applications (which typically include separate web and database interfaces), cloud-native apps take advantage of a microservices architecture that is highly distributed and dependent on potentially thousands of independent services.

Containerized microservices are small, independent runtimes that benefit from fast startups rather than from the long running-process optimizations in a traditional just-in-time (JIT) compiler.

GraalVM Native Image technology precompiles applications into stand-alone executables, without the downsides mentioned earlier. Applications start up instantaneously and with lower memory requirements are ideal for small services.

High-Performance Runtime for Microservices

DevOps teams are adopting microservices architecture for greater agility, better customer experience, and lower operational costs. But a successful microservice strategy

must be able to scale quickly and use resources efficiently during peak demands.

According to research by Akamai and Dynatrace, 40 percent of consumers abandon web pages and shopping carts if the response time is more than 3 seconds.

GraalVM's Native Image compiles Java applications into native platform executables. These executables start in mere milliseconds (see Figure 3-1) because the application code, dependencies, and all required runtime libraries are precompiled and linked into one native binary that is smaller and easier to deploy.

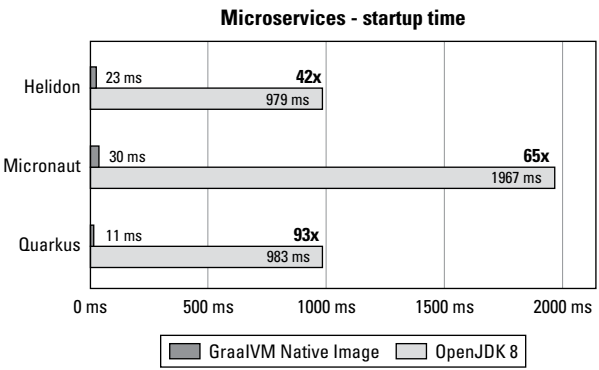

FIGURE 3-1: Startup time of microservices with GraalVM Native Image versus OpenJDK 8.

Many modern application frameworks in the Java ecosystem are compatible with GraalVM Native Image including Helidon, Micronaut, Quarkus, and (soon) Spring Boot.

Applications running on GraalVM Native Image also have a much smaller memory footprint because the Java Virtual Machine (JVM) and JIT compiler don't have to be distributed (see Figure 3-2).

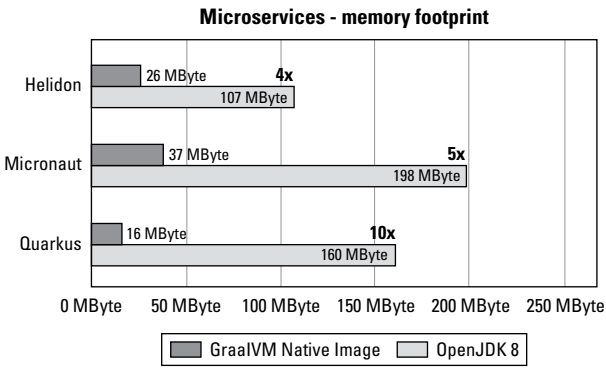

FIGURE 3-2: Memory footprint of microservices with GraalVM Native Image versus OpenJDK 8.

Very often, the distribution package size is also improved because the Java application classes, JDK runtime library classes, and dependencies are included in the compiled stand-alone native binary. It is, thus, smaller and easier to deliver.

Transforming the Economics of Cloud Deployments

The key to containing IT costs in the cloud is efficient resource CPU and memory consumption. The on-demand availability of abundant compute resources in the cloud makes it too easy for infrastructure and operations (I&O) teams, application owners, and DevOps teams to just "throw more CPUs and memory" at an application performance issue.

Although this approach may address the problem in the short term, it can lead to escalating cloud costs. For Java applications, GraalVM Native Image technology offers the performance profile of native applications, without needing to rewrite everything. Besides the ability to turn Java applications into native executables that are much better for cloud microservices workloads, GraalVM improves generated machine code to match the peak throughput performance of a JIT compiled application.

IN THIS CHAPTER

- Understanding the challenges of multi-language support
- Blossoming with the GraalVM Language Implementation Framework
- Using the right language or library for any task

Chapter 4

Enabling Multi-Language Support

In this chapter, you discover how GraalVM increases developer productivity and opens multiple language ecosystems to your applications.

The Programming Tower of Babel

The popular TIOBE Index (`www.tiobe.com/tiobe-index/programming-languages-definition`) lists almost 300 programming languages ideal for different developer needs. For example, Java is widely used for systems that require high performance and security, Python and R offer great libraries for machine learning and statistics, and JavaScript is the language of choice for many front-end developers. But these choices pose IT challenges:

- How can you be sure the technology stack you choose today will remain relevant in the future?
- What if business requirements change or technology choices limit your options?
- How can you migrate to another language?

These are a few reasons why organizations use several programming languages in a single application. But supporting a polyglot application architecture is hard because such applications are difficult to write and maintain, and suffer from some interlanguage performance overhead.

One Virtual Machine (VM) to Rule Them All

GraalVM's Language Implementation Framework (codename "Truffle") supports the execution of high-performance language implementations on the Java Virtual Machine (JVM). Because all languages are running on the same platform, there's no performance penalty for using multiple languages in the same application. Data can be passed between languages at no cost.

The benefits of GraalVM's multi-language support include the following:

- **Optimizations:** GraalVM languages run with high performance and require fewer resources using a modern optimized compiler.
- **Interoperability:** Interactions between each language are resolved by GraalVM, taking the burden off developers.
- **Common tooling:** GraalVM offers common tooling for applications leveraging multiple languages to increase developer productivity.
- **Security:** GraalVM's sandbox executes less trusted application code, libraries, and scripts with greater control by limiting access to data, memory, threads, file system, environment, and CPU time.

GraalVM can run the following languages:

- **Java and JVM languages:** Scala, Kotlin, Groovy, Clojure, and others. JVM languages can be run in a context of Java Hotspot VM or compiled as GraalVM native images.
- **JavaScript:** GraalVM includes an ECMAScript 2020 compliant JavaScript runtime, supporting high-performance for JavaScript.
- **Ruby:** GraalVM runs Ruby code with high language standard compatibility and better performance for CPU-intensive workloads.
- **Python:** GraalVM executes Python code and other languages with greater interoperability.
- **R:** Allows GraalVM to work on statistics and data science tasks in an efficient manner.
- **Low-level virtual machine (LLVM) languages:** C, C++, Rust, and others can be executed on GraalVM.

The development of polyglot applications is simplified by using GraalVM's Visual Studio Code extension for easy program editing and debugging across supported languages. Figure 4-1 shows an example of how easy it is use Java and Python together.

```java
import org.graalvm.polyglot.*;
import org.graalvm.polyglot.proxy.*;

public class HelloPolyglot {
    public static void main(String[] args) {
        System.out.println("Hello Java!");
        try (Context context = Context.create()) {
            context.eval("python", "print('Hello Python!')");
        }
    }
}
```

FIGURE 4-1: Sample code for Java and Python interoperability.

The developer just imports a few dependencies and specifies the guest language to write a polyglot application. The application then runs as a normal Java application.

Here are some examples of how the rich capabilities of GraalVM improve real-world applications:

- **Bringing the power of graphics processing units (GPUs) to modern programming languages with NVIDIA:** GPUs are great for compute-intensive workloads but utilizing them with modern high-level languages isn't easy. To resolve this, NVIDIA and Oracle Labs developed grCUDA, an open-source language binding that enables efficient data exchange between a host language and existing GPU kernels. You can learn more at `https://github.com/NVIDIA/grcuda`.

- **Optimizing machine learning performance at NetSuite with Python:** NetSuite is a cloud enterprise resource planning (ERP) application that provides business management services for thousands of organizations worldwide. The NetSuite engineering team has been working on its next-generation recommendation system and used Python and grCUDA on GraalVM to build fast and highly accurate machine learning models within their existing Java application. In this way, an existing Java system can benefit from a rich Python ML ecosystem and GPU acceleration.

- **React.js server-side rendering:** Many web applications use a client-side framework such as React.js for a rich and modern user experience. But this slows down page loads. A solution is to render pages on the server and provide ready-to-view HTML content to the client. For example, if your backend is written in Java or Scala, you don't need to rewrite code; GraalVM can run both Java and JavaScript. Learn how a social network service, ok.ru, used GraalVM for this use case (`https://prog.world/new-odnoklassniki-frontend-launching-react-in-java-part-i`).

IN THIS CHAPTER

- Increasing speed and reducing latency
- Establishing a foundation for modern applications
- Promoting interoperability among different languages
- Powering the cloud
- Improving security
- Lighting up Apache Spark

Chapter 5

Exploring GraalVM Use Cases

In this chapter, you explore real-world business use cases for GraalVM, from accelerating application performance to enabling embedded applications and more.

Accelerating Application Performance

The practical value of GraalVM's high-performance compiler and Native Image utility is to help companies meet their service-level agreement (SLA) commitments to customers and partners. Improved performance and reduced resource utilization save money by using fewer servers or paying for fewer cloud virtual machine hours each month to do the same amount of work as before.

TIP

Oracle WebLogic and Oracle Coherence are certified on GraalVM Enterprise to provide a fully supported platform for high performance Java Enterprise Edition (EE) applications.

SOMETHING TO TWEET ABOUT

The Twitter tweet service, along with 30 to 40 of its other core services (including Social and News), runs entirely on GraalVM in production. Twitter can run about 10 percent more load on the same infrastructure, resulting in substantial savings. According to Twitter's Chris Thalinger, "We save a lot of money by using it." Learn more at `https://youtu.be/jLnedMcXYEs?t=3`.

TIP

Read Chapter 2 about GraalVM compiler optimizations that improve app performance.

Building a Java Microservices Foundation

With the growing adoption of microservice architectures, developers struggle with how to create Java microservices that compete with other implementation choices like Go, Node.js, and even Python. Microservices should be as small as possible to support rapid scale up, start up quickly, and consume as few resources as possible. Java applications are not typically known for this.

TIP

GraalVM Native Image is ideal for microservices and allows developers to use their Java skills and leverage the huge Java ecosystem. Microservices development is even easier using GraalVM Native Image supported by leading microservice frameworks like Helidon, Micronaut, and Quarkus, as well as with Spring.

TIP

Read Chapter 3 and discover why GraalVM is the ideal microservices platform.

ALIBABA SAYS "OPEN SESAME" TO GRAALVM

Alibaba, with 10,000 Java developers and more than a billion (yes, billion) lines of Java code needed to choose a technology for building scalable Spring Boot applications on Alibaba Cloud. They were worried about using Java because some developers believe that Java would boot slowly and use too much memory. Fortunately, GraalVM Native Image provides a way to deploy Java applications that start instantly, using a smaller memory footprint.

By using GraalVM Native Image, Alibaba dramatically improved the performance of its Java microservices including:

- One hundred times faster startup time
- Eighty-three percent reduction in memory usage
- Ninety-three percent reduction in garbage collection pause time
- Eighty percent reduction in the cost to their customers

Learn more at `https://medium.com/graalvm/static-compilation-of-java-applications-at-alibaba-at-scale-2944163c92e`.

Supporting Multiple Languages

Developers use different programming languages to solve different kinds of problems. For example, Java is an excellent language for building modern structured enterprise applications, R is ideal for statistical analysis, and Python is popular with data scientists for machine learning. GraalVM provides support for all these languages and many others including Ruby, JavaScript, C, C++, and FORTRAN (via Low-Level Virtual Machine, or LLVM, bitcode). One runtime that supports multiple languages is interesting, but GraalVM adds two unique features: the ability to efficiently mix multiple languages in a single polyglot application, and execution speeds that typically exceed those of the original language runtimes.

Oracle NetSuite uses GraalVM to provide customization and scripting capabilities to its end users. Using GraalVM's sandbox to run less trusted code, NetSuite can control how user-defined JavaScript scripts interact with its Java-based platform. The use of lightweight scripting languages like JavaScript and Python to customize, configure, or extend applications written in Java or Scala is a common use of GraalVM's multilanguage support. Learn more at `www.netsuite.com/blog/graal-runtime-technology-improves-netsuite-platform-developer-productivity`.

ENABLING HIGH-SPEED PURSUIT (OF APPS) FOR THE DUTCH POLICE

The Dutch Police use GraalVM's multilanguage support to integrate Scala applications with analytics written in R. This makes it possible for their data scientists to work in R, a language that they are familiar with, and for their work to be easily incorporated into Scala-based Kafka clients written by a different set of developers. GraalVM enables the Dutch Police's data scientist to work efficiently with its software development colleagues to digitize the Dutch criminal justice system. Learn more at `https://youtu.be/poNIwZjoYjs`.

TIP

Read Chapter 4 and learn how GraalVM multilanguage support increases productivity.

Leveraging Oracle Cloud Infrastructure

Cloud platforms like Oracle Cloud Infrastructure (OCI) provide elastic compute resources that enable services to scale up or down to match demand. One of the key

benefits of cloud platforms is that you only pay for what you use. By minimizing compute resource usage, you can lower your monthly cloud bill. Applications running on GraalVM run significantly faster while using less memory, which makes it a great way to shrink your cloud bill. On OCI you can take advantage of GraalVM to reduce your costs at no additional charge. GraalVM supports these deployment scenarios:

- Bare-metal and virtual machines (OCI Compute)
- Containers (Oracle Container Engine for Kubernetes)
- Functions (Oracle Functions)

The improved performance and reduced resource requirements of GraalVM can create significant cost savings for OCI customers. Not only can GraalVM lower your operations cost, but OCI itself runs services on GraalVM; systems monitoring shows a 25 percent reduction of garbage collection pause time and a 10 percent increase in the number of transactions per second, which translates to a savings to users.

REMEMBER

Many of the OCI services you're already using are powered by GraalVM, and all OCI users can also take advantage of it at no additional cost to improve the performance of their workloads and reduce their cloud resource consumption.

Reducing Risk with Native Image

GraalVM Enterprise's Native Image utility greatly reduces the attack surface of an application by removing unused code. It analyzes an application to determine which classes, methods, and fields are needed for correct execution and only compiles what's necessary into the generated native executable. A Java application's classpath includes thousands of classes from dependent libraries and the Java Development Kit (JDK). The JDK alone has included more than 4,000 classes since JDK 7, but a typical application only needs a fraction of them. GraalVM Native Image can remove as much as 96 percent of the Java code from the JDK and other libraries from a generated native executable. With fewer classes, there are fewer potential vulnerabilities for an attacker to exploit.

Faster Streaming Infrastructure with Spark and Kafka

GraalVM shows outstanding results on applications and frameworks that are cornerstones of modern enterprise infrastructure. For example, Apache Spark is a popular analytics engine for big data processing that runs on the

JVM. Looking at the benchmarks for Spark in the Renaissance benchmark suite, GraalVM shows 1.6 times better performance compared to OpenJDK 8. Go to `https://blogs.oracle.com/graalvm/apache-spark%e2%80%94lightning-fast-on-graalvm-enterprise` to learn more.

Similarly, Apache Kafka — a streaming platform that has grown ever more popular lately — runs on the JVM and can show better performance when used with GraalVM. What's most interesting is that in sample experiments, Kafka server shows better throughput and improved tail latencies for processing incoming messages.

IN THIS CHAPTER

- Leveraging proven experience
- Building on open-source GraalVM and popular frameworks
- Optimizing and modernizing application workloads
- Controlling costs and securing your application workloads
- Innovating for the future

Chapter 6

Ten Reasons to Use GraalVM

Here are ten reasons you should consider GraalVM for your enterprise applications:

» **It's backed by ten years of research by Oracle Labs.** GraalVM represents more than a decade of

research, patented optimizations, and virtual machine technology that provides significant application performance and efficiency benefits.

- **It's built on the Oracle-led GraalVM open-source project** (`https://graalvm.org`). The community includes participants from leading companies and an advisory board with members from industry and academia.
- **It's built on Oracle Java SE.** Java continues to be the leading application development language with 98 percent of the Fortune 100 running Java and 45 billion Java Virtual Machines (JVMs) globally. Enterprises rely on Oracle Java SE for reliability, security, and regular updates. Although GraalVM Enterprise provides significant benefits for Java applications, it's built on proven Oracle Java SE.
- **It enables high-performance Java.** The just-in-time (JIT) compiler includes only the code needed to execute programs. Applications written in multiple programming languages can run up to 50 percent faster on GraalVM.
- **It transformed and revolutionized the cloud deployment economics of Java.** GraalVM enables applications and microservices to start up instantly in multi-cloud environments. The ahead-of-time (AOT) compiler helps applications spend less time doing garbage collection. This means better

application performance and lower cloud operation costs. It also enables a multi-cloud architecture that can support hybrid workloads with proven deployments on all major public clouds and demonstrated consistency and predictability of workload performance on different containers and cloud instances.

- **It's supported by all major microservices frameworks.** These frameworks include Helidon, Micronaut, Quarkus, Spring Boot, and Tomcat.
- **It enhances security.** GraalVM Native Image utility reduces application vulnerabilities. It includes only the Java code required to execute the application, significantly reducing the attack surface. There is no dynamic class loading that can be exploited in a de-serialization attack. An AOT compiled native executable won't be impacted by a vulnerability that may exist in a class that's part of a Java library that the application included but never actually used. So, the application won't need to be patched as frequently to address vulnerabilities in third-party dependencies.
- **It offers reduced application vulnerability**. The GraalVM Native Image utility ahead-of-time (AOT) compiler only includes the Java code required to execute the application, significantly reducing the attack surface.

- **It's at the center of innovation.** Modernize large enterprise applications and simplify microservice deployments. The unique set of capabilities that GraalVM provides has led to its being integrated into Oracle NetSuite for user scripting. Oracle Coherence used GraalVM to bring advanced data grid and in-memory caching technology to JavaScript and Node.js, and other languages. GraalVM innovations are influencing specification updates like the recent announcement of the OpenJDK's Project Leyden.
- **It has been integrated in Oracle Database 21 to provide multilingual support for stored procedures to improve its usability for modern full-stack developers.**

Deploy to a modern high-performance runtime to accelerate your applications

GraalVM is a high-performance runtime for deploying applications on premises and in the cloud. GraalVM significantly accelerates application performance by optimizing CPU and memory usage, enabling you to reduce costs by using existing infrastructure and cloud resources more efficiently. It's the ideal platform for microservices and for modernizing traditional business applications without rewriting code.

Inside...

- Leverage new compiler optimizations
- Enable faster workloads
- Increase performance at scale
- Deploy microservices efficiently
- Leverage polyglot support
- Improve developer productivity
- Reduce your app attack surface

ORACLE

Go to Dummies.com™
for videos, step-by-step photos, how-to articles, or to shop!

ISBN: 978-1-119-76642-1
Not for resale

for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.