Extracted from:

Modern Systems Programming with Scala Native

Write Lean, High-Performance Code without the JVM

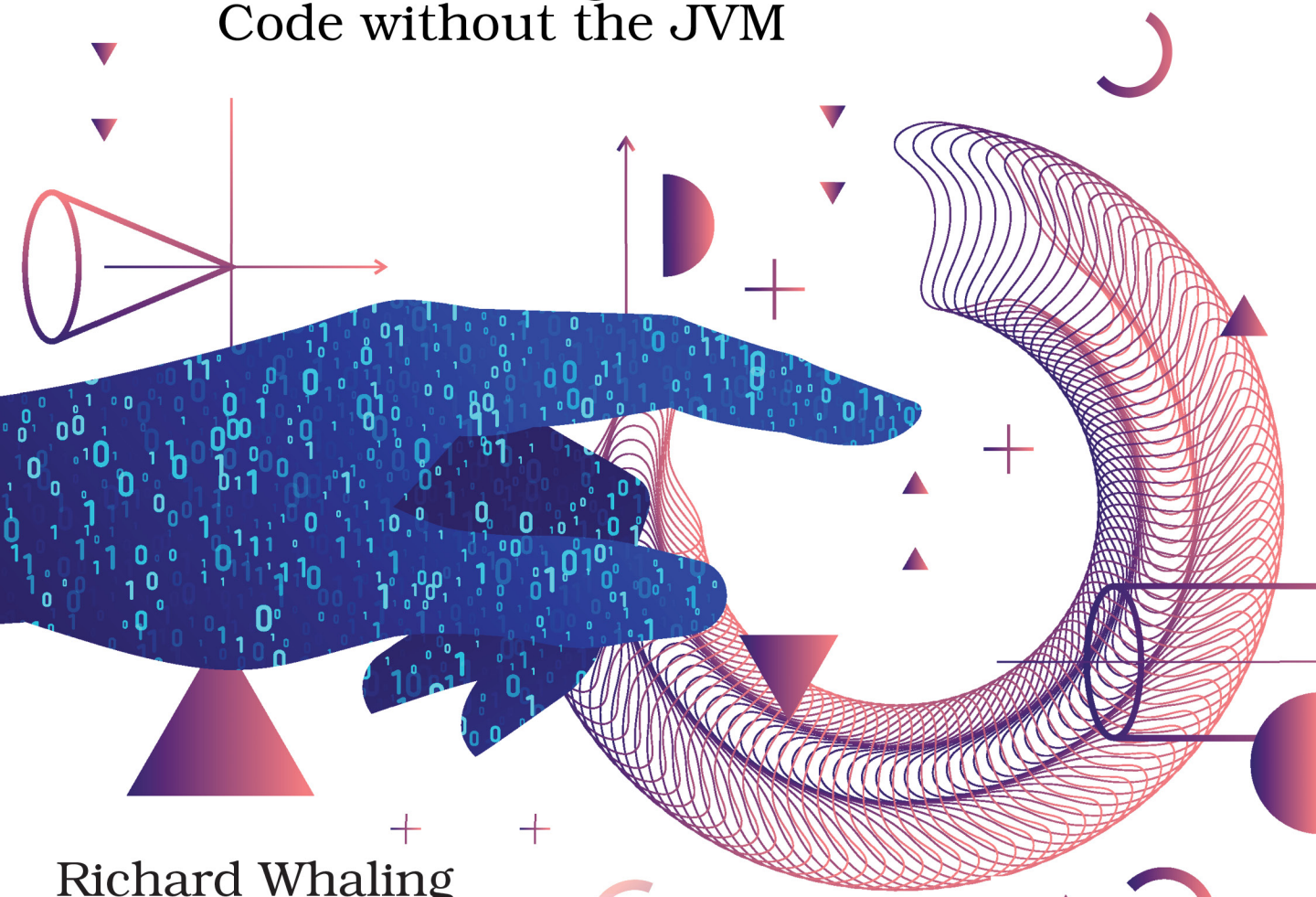
The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Modern Systems Programming with Scala Native

Write Lean, High-Performance Code without the JVM



Richard Whaling
Foreword by Denys Shabalin, Author of Scala Native
Edited by Katharine Dvorak

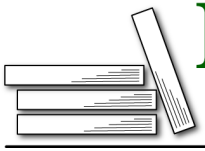
Modern Systems Programming with Scala Native

Write Lean, High-Performance Code without the JVM

Richard Whaling

The Pragmatic Bookshelf

Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Executive Editor: Dave Rankin
Development Editor: Katharine Dvorak
Copy Editor: L. Sakhi MacMillan
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-622-8
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—January 2020

Preface

If you've ever been frustrated by the many layers of abstraction between your code and the machine it runs on, you're looking at the right book. Over the coming chapters, I'll show you how you can use Scala Native to build efficient, modern programs from the ground up, focusing on practical use cases like REST clients, microservices, and bulk data processing. With Scala Native, you don't have to choose between elegant code and bare-metal performance.

Who This Book Is For

This book is for anyone who wants to learn how to build real software from scratch with a cutting-edge language. Maybe you learned Scala or Clojure on the job, but want to learn more about how to work "close to the metal" in a functional language. Maybe you're an enthusiast and want to write smaller, lightweight Scala programs that perform on tiny, near-embedded Linux systems. Or maybe you're a devops engineer with a strong Java background, who is just learning Scala, and you want to write strongly typed, testable code that doesn't impose the runtime penalties of the JVM. In other words, this book is for the folks who are my peers and colleagues in the Scala and greater JVM-language community.

I've tried my hardest to make this book accessible to folks with no prior systems programming experience—you'll learn about arrays, pointers, and the rest, as we go along.

All the code is in Scala, but we won't be using the advanced Scala techniques you might find in a functional programming text. When we do use intermediate-level techniques like implicits, I'll call them out.

That said, a few days' worth of experience with Scala is highly recommended. If you're totally new to Scala, there are a lot of great resources online. The official Tour of Scala[1] is a great place to start, and if you want to go deeper,

1. https://docs.scala-lang.org/tour/tour-of-scala.html

Dave Gurnell's and Noel Welsh's *Creative Scala*[2] or Martin Odersky's Functional Programming Principles in Scala[3] online course are both excellent resources. *Pragmatic Scala*[4] by Venkat Subramaniam offers a great, approachable book-length treatment, as does *Scala for the Impatient*[5] by Cay Horstmann. *Programming in Scala*,[6] by Martin Odersky, Lex Spoon, and Bill Venners, is the official book by the author of the language, and is a great, thorough reference guide, but make sure you get the third edition—the second and first editions are significantly out of date now.

What's In This Book

This book is designed as a series of projects that introduce the fundamental and powerful techniques of systems programming, one by one. Each chapter discusses an important topic in systems programming, and in the spirit of adventure, we may not always take the most direct route to our destination. Sometimes it's best to make a few mistakes, or do things by hand a few times before we skip ahead to the "right answer."

The work will all pay off, though. As you progress and master more and more techniques, you will gradually put the pieces together into something greater than the sum of its parts. And by the end, not only will you have the code for a lightweight, asynchronous microservice framework, you'll also be able to write one yourself if you don't like the way I did it.

The book is divided into two parts:

Part I dives into the fundamental techniques of systems programming using the basic facilities that UNIX-based operating systems have had since the 1980s. Unlike traditional systems programming books, however, networking is introduced early. For a modern programmer, working with remote services over HTTP is more relevant and practical than local file I/O. I then introduce process-based concurrency and parallelism. Although most Scala programmers will be more familiar with threads, processes are a powerful technique that distinguishes Scala Native from most other programming languages, and they're a great, safe introduction to asynchronous programming. We then look at combining these techniques to build a minimalist HTTP server and

2. https://www.creativescala.org/creative-scala.html
3. https://www.coursera.org/learn/progfun1
4. https://pragprog.com/book/vsscala2/pragmatic-scala
5. https://horstmann.com/scala
6. https://www.artima.com/shop/programming_in_scala_3ed

measure its performance with a simple stress test. However, we also look at the limits of these traditional techniques.

In Part II, we'll put the "modern" in "modern systems programming." From this point on, all of our code will be fully asynchronous, building upon the capabilities of the event loop library, libuv. Working with an industrial-strength C library like this, we'll introduce new complexities to our code, but it also gives Scala Native a chance to truly shine. With libuv, we'll revisit our HTTP server, introduce idiomatic Scala concurrency techniques, and learn how to work with durable data stores. Then, when we put those components together, we'll have built a framework for solving real-world problems. I'm skeptical of buzzwords, but the low overhead and light footprint of Scala Native code really does put JVM-based "microservices" to shame.

Working with the Code

All the code in this book has been tested with Scala Native 0.4.0-M2, Scala 2.11.12, and sbt 0.13.15 on Mac OS X and Linux (via Docker). You can get detailed instructions on how to set up a Scala Native development environment for Mac, Windows, or Linux in Appendix 1, Setting Up the Environment, on page ?. Since slight environment differences can be disruptive to low-level programs, I recommend using Docker on your preferred development machine, and the examples in the text reflect that.

A Note on Versions

Scala Native is rapidly evolving. The example code in this chapter, as well as all other code in this book, is written for the most recent version of Scala Native available, 0.4.0-M2. To ensure forward compatibility, all of the sbt projects include a compatibility shim file; to use with a newer version, just remove the shim!

You may also notice all the code is for Scala 2.11. When Scala 2.12 and 2.13 become available for Scala Native, I'll update the code files as well. You can download the latest version of the sample code on the pragprog.com website (https://pragprog.com/book/rwscala).

How the Code Is Organized

You can download the source code used in this book from the book's web page at pragprog.com.[7] If you're reading the electronic version of this book,

7. https://pragprog.com/titles/rwscala/source_code

you can click the box above the code excerpts to download that source code directly. Or, if you use the Docker environment, you've already got it.

The code is organized by chapter, and within each chapter, the code is organized into individual projects, each with its own folder. Each project is a self-contained codebase, designed to be built by sbt,[8] the standard Scala build tool.

One important note if you're trying to modify the code: sometimes, for concise presentation, I will not show import statements and outer `object Main` wrappers in the code printed in the book. For example, have a look at this snippet:

```
import scalanative.unsafe._

object Main {
  def main(args:Array[String]:Unit = {
    // invoking various functions here
    ???
  }

  def helperFunction(arg1:Int, arg2:String) = ???
}
```

It may be displayed as:

```
def main(args:Array[String]:Unit = {
  // invoking various functions here
  ???
}

def helperFunction(arg1:Int, arg2:String) = ???
```

However, all the code files that you download and use are fully functional and complete. If you're interested in modifying my examples, definitely start with the files.

About Text Editors

Although there's no one standard for text editors when it comes to Scala, there are many options. Because Scala Native is still relatively new, the support for it is imperfect in complex IDEs. I instead recommend a plain text editor with good support for Scala syntax, like VSCode, Atom, emacs, or vi. Part of the beauty of starting from scratch is that we don't have to deal with giant Java dependencies with hundreds of classes—our code will be lean enough to write without sophisticated editor assistance. That said, I've found that the Metals plugin[9] for VSCode offers a good balance of common-sense help in an unobtrusive way.

8. https://www.scala-sbt.org
9. https://scalameta.org/metals

Online Resources

You'll definitely want to keep tabs on the book's web page at pragprog.com[10] for all the latest code and updates. And if you find any errata, there's a place to let me know about it.[11] I've also created a dedicated chat room on Gitter.[12] If you have any problems building or running the code in the book, or just want to hang out and chat, come on by! I also highly recommend perusing Scala Native's official site,[13] and referring to the Scala Native source code on Github[14] for the occasional deep dive.

With those resources in hand, it's time to get started!

10. https://pragprog.com/book/rwscala
11. https://pragprog.com/titles/rwscala/errata
12. https://gitter.im/scala-native-book/community
13. https://scala-native.readthedocs.io
14. https://github.com/scala-native/scala-native