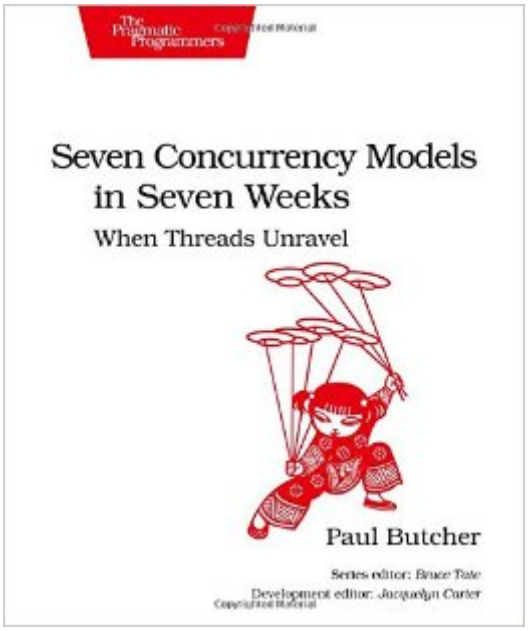
Seven Concurrency Models In Seven Weeks: When Threads Unravel (The Pragmatic Programmers)



Synopsis

Your software needs to leverage multiple cores, handle thousands of users and terabytes of data, and continue working in the face of both hardware and software failure. Concurrency and parallelism are the keys, and Seven Concurrency Models in Seven Weeks equips you for this new world. See how emerging technologies such as actors and functional programming address issues with traditional threads and locks development. Learn how to exploit the parallelism in your computer's GPU and leverage clusters of machines with MapReduce and Stream Processing. And do it all with the confidence that comes from using tools that help you write crystal clear, high-quality code.This book will show you how to exploit different parallel architectures to improve your code's performance, scalability, and resilience. You'll learn about seven concurrency models: threads and locks, functional programming, separating identity and state, actors, sequential processes, data parallelism, and the lambda architecture.Learn about the perils of traditional threads and locks programming and how to overcome them through careful design and by working with the standard library. See how actors enable software running on geographically distributed computers to collaborate, handle failure, and create systems that stay up 24/7/365. Understand why shared mutable state is the enemy of robust concurrent code, and see how functional programming together with technologies such as Software Transactional Memory (STM) and automatic parallelism help you tame it.You'll learn about the untapped potential within every GPU and how GPGPU software can unleash it. You'll see how to use MapReduce to harness massive clusters to solve previously intractable problems, and how, in concert with Stream Processing, big data can be tamed. With an understanding of the strengths and weaknesses of each of the different models and hardware architectures, you'll be empowered to tackle any problem with confidence.What You Need:The example code can be compiled and executed on *nix, OS X, or Windows. Instructions on how to download the supporting build systems are given in each chapter.

Book Information

Series: The Pragmatic Programmers
Paperback: 300 pages
Publisher: Pragmatic Bookshelf; 1 edition (July 10, 2014)
Language: English
ISBN-10: 1937785653
ISBN-13: 978-1937785659
Product Dimensions: 7.5 x 0.6 x 9.2 inches

Shipping Weight: 1.2 pounds (View shipping rates and policies)
Average Customer Review: 4.5 out of 5 starsÂ Â See all reviewsÂ (25 customer reviews)
Best Sellers Rank: #355,609 in Books (See Top 100 in Books) #40 inÂ Books > Computers & Technology > Programming > Parallel Programming #106 inÂ Books > Computers & Technology > Computer Science > Information Theory #1532 inÂ Books > Textbooks > Computer Science > Programming Languages

Customer Reviews

It is a little different from what the title suggests - it is supposed to be about concurrency models, and it actually is, but it covers more about Clojure than concurrency models themselves. In other words, this book is mainly about how to use the concurrency models in Clojure (and why some features are/aren't included in Clojure itself). If you are familiar with Clojure, this book would be the best. If you are not, however, you can feel uneasy in many parts of this book, even if you are interested in studying it - though this book covers many features of Clojure, it is not enogh to learn Clojure with this book alone.

More than likely you're browsing here and reading reviews because you have some practical experience in software development. And likely that means you've had to slog through some ponderous textbooks in the past, either while at university or when your job has tasked you with a need for some knowledge. Thankfully, this book will never give you flashbacks of those times falling asleep face first into a small puddle of drool in the third chapter of some dusty textbook recommended by your professor.This is a well-paced, smart, engaging....fun?? (can I say fun in relation to learning?) book that made me feel like I was hanging out with a brilliant guy for a few hours each week who can "talk the talk and walk the walk", so to speak.There is balance to each of the modules: theory, practice, and self-learning, and there is a tone set very early in the book: (1) here is an idea and a practical analogy, (2) here is what will be taught in the following few chapters, (3) here is a wrap-up with strengths and weaknesses, and (4) here is where you can find more information on this and other relatable topics. In between there are break-out discussions, there are real code samples that make sense(!), there are self-learning suggestions, and there are basic hands-on exercises that (almost) anyone can do.And all of this is done with a steady, easy pace, mapped out (if you prefer) in calendar time so that its not overwhelming and fits into the busy lifestyles of the very people that should be reading this book...now!I learned a lot, more than I thought actually, especially because of the use of languages like Clojure and Elixir, which gave me

a view of the world of concurrent programming that I could never hope to find with basic Javascript and C. Read the book. Embrace the models. Do the exercises. You'll be a stronger and more intelligent programmer in a matter of (seven) weeks because of it.

I work in communications software and have experience of concurrent programming through Erlang, but needed to extend my knowledge out to other areas of real-time processing. This book was fantastic for going through the different approaches, the pros and cons and gotchas along with insights into low level things that go on inside the VM and operating systems. It's easy to read, but also stretches you. Great book.

An excellent book for experienced software engineers and newcomers alike. The examples are detailed enough to serve their purpose without getting too bogged down with implementation details which would slow the pace of the book. The concurrency models are explained clearly and concisely. Personally I would have preferred the CSP chapters to use Go (they use Clojure with core.async) but this is a book about concurrency not languages so that is irrelevant.If you are looking for an overview of the tools and techniques available now for building concurrent systems this book is for you.

This book is a concise and clear picture of the major options available to solve problems of concurrency. Today if you want to understand computing solutions that tackle the toughest problems you need a clear picture of what really work. This book is an excellent navigator regarding the choices programmers faces at the beginning of inspiring data projects.

This is a solid survey of different tools to approach concurrency. It is VERY good at that, but once you've chosen a specific toolkit for whatever problem you are solving now, there are going to be other resources that can help you get deeper into the specifics of any particular model. So I would recommend this book both to anyone beginning with concurrent and/or parallel programming, but also to more "expert" programmers who might already know a couple of the models in depth but want to learn concepts they might borrow from the others---specific languages may, for example, have built-in support to favor specific models, but they can all be adapted to your own environment. So even if you feel comfortable using Java's concurrency tools, for example, you may want to implement MapReduce or Actors for yourself, even though those are "most commonly" associated with Hadoop or Erlang, and so Paul's examples use those. Doing so keeps his example small and

focused on the use of the models, but knowing about the model lets you use its ideas in whatever your work environment.

This is a great book if you want to surf through different concurrency models available at your hand via various programming languages/libraries. The point isn't really to learn the languages/tools, but it certainly helps to be able to read some real world code when trying to grasp the various models.The book tries hard to be more pragmatic than pedantic. It is an easy read that tries to distill the massive amounts of research done in the field.