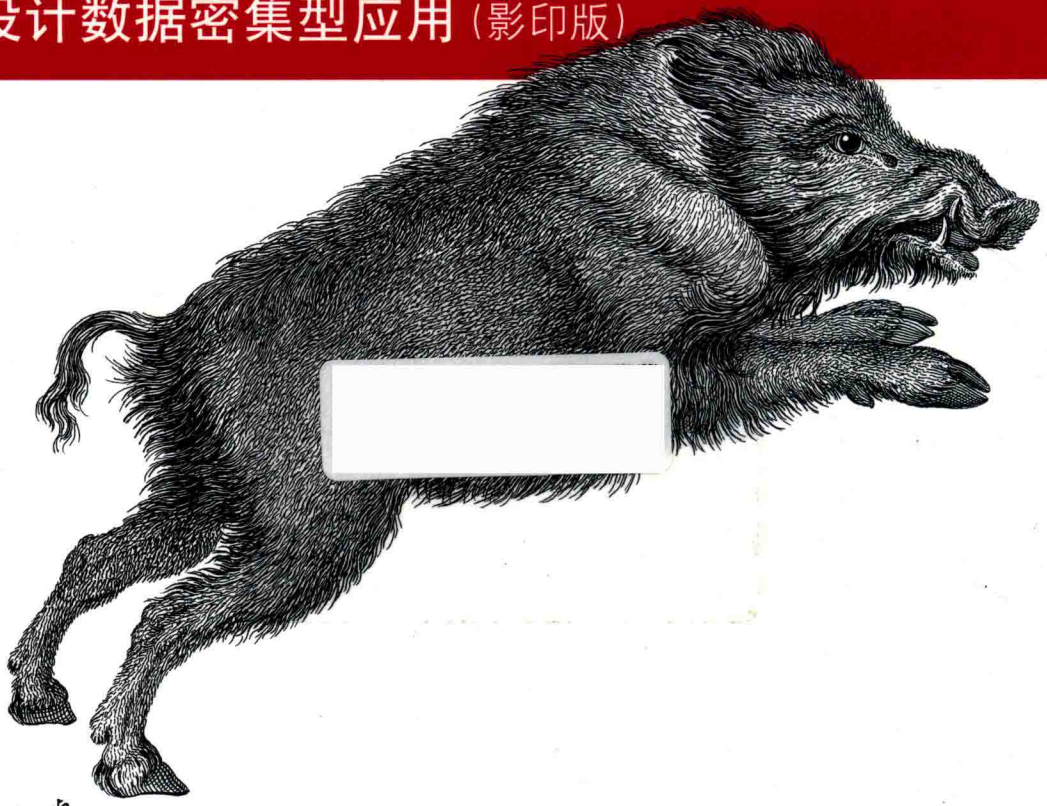
O'REILLY®

Designing Data-Intensive Applications

设计数据密集型应用（影印版）



東南大學出版社

Martin Kleppmann 著

设计数据密集型应用（影印版）

Designing Data-Intensive Applications

Martin Kleppmann 著

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc.授权东南大学出版社出版

南京 东南大学出版社

图书在版编目(CIP)数据

设计数据密集型应用:英文/(英)马丁·科勒普曼(Martin Kleppmann)著. —影印本. —南京:东南大学出版社,2017.10

书名原文:Designing Data - Intensive Applications

ISBN 978 - 7 - 5641 - 7385 - 2

Ⅰ.①设… Ⅱ.①马… Ⅲ.①软件工具—基本知识—英文 Ⅳ.①TP311.56

中国版本图书馆 CIP 数据核字(2017)第 197051 号

图字:10 - 2017 - 343 号

© 2017 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2017. Authorized reprint of the original English edition, 2017 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc.出版 2017。

英文影印版由东南大学出版社出版 2017。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc.的许可。

版权所有,未得书面许可,本书的任何部分和全部不得以任何形式重制。

设计数据密集型应用(影印版)

出版发行:东南大学出版社

地　　址:南京四牌楼 2 号　　邮编:210096

出 版 人:江建中

网　　址:http://www.seupress.com

电子邮件:press@seupress.com

印　　刷:常州市武进第三印刷有限公司

开　　本:787 毫米×980 毫米　　16 开本

印　　张:38.5

字　　数:754 千字

版　　次:2017 年 10 月第 1 版

印　　次:2017 年 10 月第 1 次印刷

书　　号:ISBN 978 - 7 - 5641 - 7385 - 2

定　　价:99.00 元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025 - 83791830

Preface

If you have worked in software engineering in recent years, especially in server-side and backend systems, you have probably been bombarded with a plethora of buzzwords relating to storage and processing of data. NoSQL! Big Data! Web-scale! Sharding! Eventual consistency! ACID! CAP theorem! Cloud services! MapReduce! Real-time!

In the last decade we have seen many interesting developments in databases, in distributed systems, and in the ways we build applications on top of them. There are various driving forces for these developments:

- Internet companies such as Google, Yahoo!, Amazon, Facebook, LinkedIn, Microsoft, and Twitter are handling huge volumes of data and traffic, forcing them to create new tools that enable them to efficiently handle such scale.
- Businesses need to be agile, test hypotheses cheaply, and respond quickly to new market insights by keeping development cycles short and data models flexible.
- Free and open source software has become very successful and is now preferred to commercial or bespoke in-house software in many environments.
- CPU clock speeds are barely increasing, but multi-core processors are standard, and networks are getting faster. This means parallelism is only going to increase.
- Even if you work on a small team, you can now build systems that are distributed across many machines and even multiple geographic regions, thanks to infrastructure as a service (IaaS) such as Amazon Web Services.
- Many services are now expected to be highly available; extended downtime due to outages or maintenance is becoming increasingly unacceptable.

Data-intensive applications are pushing the boundaries of what is possible by making use of these technological developments. We call an application *data-intensive* if data is its primary challenge—the quantity of data, the complexity of data, or the speed at

which it is changing—as opposed to *compute-intensive*, where CPU cycles are the bottleneck.

The tools and technologies that help data-intensive applications store and process data have been rapidly adapting to these changes. New types of database systems ("NoSQL") have been getting lots of attention, but message queues, caches, search indexes, frameworks for batch and stream processing, and related technologies are very important too. Many applications use some combination of these.

The buzzwords that fill this space are a sign of enthusiasm for the new possibilities, which is a great thing. However, as software engineers and architects, we also need to have a technically accurate and precise understanding of the various technologies and their trade-offs if we want to build good applications. For that understanding, we have to dig deeper than buzzwords.

Fortunately, behind the rapid changes in technology, there are enduring principles that remain true, no matter which version of a particular tool you are using. If you understand those principles, you're in a position to see where each tool fits in, how to make good use of it, and how to avoid its pitfalls. That's where this book comes in.

The goal of this book is to help you navigate the diverse and fast-changing landscape of technologies for processing and storing data. This book is not a tutorial for one particular tool, nor is it a textbook full of dry theory. Instead, we will look at examples of successful data systems: technologies that form the foundation of many popular applications and that have to meet scalability, performance, and reliability requirements in production every day.

We will dig into the internals of those systems, tease apart their key algorithms, discuss their principles and the trade-offs they have to make. On this journey, we will try to find useful ways of *thinking about* data systems—not just *how* they work, but also *why* they work that way, and what questions we need to ask.

After reading this book, you will be in a great position to decide which kind of technology is appropriate for which purpose, and understand how tools can be combined to form the foundation of a good application architecture. You won't be ready to build your own database storage engine from scratch, but fortunately that is rarely necessary. You will, however, develop a good intuition for what your systems are doing under the hood so that you can reason about their behavior, make good design decisions, and track down any problems that may arise.

Who Should Read This Book?

If you develop applications that have some kind of server/backend for storing or processing data, and your applications use the internet (e.g., web applications, mobile apps, or internet-connected sensors), then this book is for you.

此为试读，需要完整PDF请访问：www.ertongbook.com

This book is for software engineers, software architects, and technical managers who love to code. It is especially relevant if you need to make decisions about the architecture of the systems you work on—for example, if you need to choose tools for solving a given problem and figure out how best to apply them. But even if you have no choice over your tools, this book will help you better understand their strengths and weaknesses.

You should have some experience building web-based applications or network services, and you should be familiar with relational databases and SQL. Any nonrelational databases and other data-related tools you know are a bonus, but not required. A general understanding of common network protocols like TCP and HTTP is helpful. Your choice of programming language or framework makes no difference for this book.

If any of the following are true for you, you'll find this book valuable:

- You want to learn how to make data systems scalable, for example, to support web or mobile apps with millions of users.
- You need to make applications highly available (minimizing downtime) and operationally robust.
- You are looking for ways of making systems easier to maintain in the long run, even as they grow and as requirements and technologies change.
- You have a natural curiosity for the way things work and want to know what goes on inside major websites and online services. This book breaks down the internals of various databases and data processing systems, and it's great fun to explore the bright thinking that went into their design.

Sometimes, when discussing scalable data systems, people make comments along the lines of, "You're not Google or Amazon. Stop worrying about scale and just use a relational database." There is truth in that statement: building for scale that you don't need is wasted effort and may lock you into an inflexible design. In effect, it is a form of premature optimization. However, it's also important to choose the right tool for the job, and different technologies each have their own strengths and weaknesses. As we shall see, relational databases are important but not the final word on dealing with data.

Scope of This Book

This book does not attempt to give detailed instructions on how to install or use specific software packages or APIs, since there is already plenty of documentation for those things. Instead we discuss the various principles and trade-offs that are fundamental to data systems, and we explore the different design decisions taken by different products.

In the ebook editions we have included links to the full text of online resources. All links were verified at the time of publication, but unfortunately links tend to break frequently due to the nature of the web. If you come across a broken link, or if you are reading a print copy of this book, you can look up references using a search engine. For academic papers, you can search for the title in Google Scholar to find open-access PDF files. Alternatively, you can find all of the references at *https://github.com/ept/ddia-references*, where we maintain up-to-date links.

We look primarily at the *architecture* of data systems and the ways they are integrated into data-intensive applications. This book doesn't have space to cover deployment, operations, security, management, and other areas—those are complex and important topics, and we wouldn't do them justice by making them superficial side notes in this book. They deserve books of their own.

Many of the technologies described in this book fall within the realm of the *Big Data* buzzword. However, the term "Big Data" is so overused and underdefined that it is not useful in a serious engineering discussion. This book uses less ambiguous terms, such as single-node versus distributed systems, or online/interactive versus offline/batch processing systems.

This book has a bias toward free and open source software (FOSS), because reading, modifying, and executing source code is a great way to understand how something works in detail. Open platforms also reduce the risk of vendor lock-in. However, where appropriate, we also discuss proprietary software (closed-source software, software as a service, or companies' in-house software that is only described in literature but not released publicly).

Outline of This Book

This book is arranged into three parts:

1. In Part I, we discuss the fundamental ideas that underpin the design of data-intensive applications. We start in Chapter 1 by discussing what we're actually trying to achieve: reliability, scalability, and maintainability; how we need to think about them; and how we can achieve them. In Chapter 2 we compare several different data models and query languages, and see how they are appropriate to different situations. In Chapter 3 we talk about storage engines: how databases arrange data on disk so that we can find it again efficiently. Chapter 4 turns to formats for data encoding (serialization) and evolution of schemas over time.
2. In Part II, we move from data stored on one machine to data that is distributed across multiple machines. This is often necessary for scalability, but brings with it a variety of unique challenges. We first discuss replication (Chapter 5), partitioning/sharding (Chapter 6), and transactions (Chapter 7). We then go into

more detail on the problems with distributed systems (Chapter 8) and what it means to achieve consistency and consensus in a distributed system (Chapter 9).

3. In Part III, we discuss systems that derive some datasets from other datasets. Derived data often occurs in heterogeneous systems: when there is no one database that can do everything well, applications need to integrate several different databases, caches, indexes, and so on. In Chapter 10 we start with a batch processing approach to derived data, and we build upon it with stream processing in Chapter 11. Finally, in Chapter 12 we put everything together and discuss approaches for building reliable, scalable, and maintainable applications in the future.

References and Further Reading

Most of what we discuss in this book has already been said elsewhere in some form or another—in conference presentations, research papers, blog posts, code, bug trackers, mailing lists, and engineering folklore. This book summarizes the most important ideas from many different sources, and it includes pointers to the original literature throughout the text. The references at the end of each chapter are a great resource if you want to explore an area in more depth, and most of them are freely available online.

O'Reilly Safari

Safari (formerly Safari Books Online) is a membership-based training and reference platform for enterprise, government, educators, and individuals.

Members have access to thousands of books, training videos, Learning Paths, interactive tutorials, and curated playlists from over 250 publishers, including O'Reilly Media, Harvard Business Review, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Adobe, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, and Course Technology, among others.

For more information, please visit *http://oreilly.com/safari*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/designing-data-intensive-apps*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

This book is an amalgamation and systematization of a large number of other people's ideas and knowledge, combining experience from both academic research and industrial practice. In computing we tend to be attracted to things that are new and shiny, but I think we have a huge amount to learn from things that have been done before. This book has over 800 references to articles, blog posts, talks, documentation, and more, and they have been an invaluable learning resource for me. I am very grateful to the authors of this material for sharing their knowledge.

I have also learned a lot from personal conversations, thanks to a large number of people who have taken the time to discuss ideas or patiently explain things to me. In particular, I would like to thank Joe Adler, Ross Anderson, Peter Bailis, Márton Balassi, Alastair Beresford, Mark Callaghan, Mat Clayton, Patrick Collison, Sean Cribbs, Shirshanka Das, Niklas Ekström, Stephan Ewen, Alan Fekete, Gyula Fóra, Camille Fournier, Andres Freund, John Garbutt, Seth Gilbert, Tom Haggett, Pat Helland, Joe Hellerstein, Jakob Homan, Heidi Howard, John Hugg, Julian Hyde, Conrad Irwin, Evan Jones, Flavio Junqueira, Jessica Kerr, Kyle Kingsbury, Jay Kreps, Carl Lerche, Nicolas Liochon, Steve Loughran, Lee Mallabone, Nathan Marz, Caitie

McCaffrey, Josie McLellan, Christopher Meiklejohn, Ian Meyers, Neha Narkhede, Neha Narula, Cathy O'Neil, Onora O'Neill, Ludovic Orban, Zoran Perkov, Julia Powles, Chris Riccomini, Henry Robinson, David Rosenthal, Jennifer Rullmann, Matthew Sackman, Martin Scholl, Amit Sela, Gwen Shapira, Greg Spurrier, Sam Stokes, Ben Stopford, Tom Stuart, Diana Vasile, Rahul Vohra, Pete Warden, and Brett Wooldridge.

Several more people have been invaluable to the writing of this book by reviewing drafts and providing feedback. For these contributions I am particularly indebted to Raul Agepati, Tyler Akidau, Mattias Andersson, Sasha Baranov, Veena Basavaraj, David Beyer, Jim Brikman, Paul Carey, Raul Castro Fernandez, Joseph Chow, Derek Elkins, Sam Elliott, Alexander Gallego, Mark Grover, Stu Halloway, Heidi Howard, Nicola Kleppmann, Stefan Kruppa, Bjorn Madsen, Sander Mak, Stefan Podkowinski, Phil Potter, Hamid Ramazani, Sam Stokes, and Ben Summers. Of course, I take all responsibility for any remaining errors or unpalatable opinions in this book.

For helping this book become real, and for their patience with my slow writing and unusual requests, I am grateful to my editors Marie Beaugureau, Mike Loukides, Ann Spencer, and all the team at O'Reilly. For helping find the right words, I thank Rachel Head. For giving me the time and freedom to write in spite of other work commitments, I thank Alastair Beresford, Susan Goodhue, Neha Narkhede, and Kevin Scott.

Very special thanks are due to Shabbir Diwan and Edie Freedman, who illustrated with great care the maps that accompany the chapters. It's wonderful that they took on the unconventional idea of creating maps, and made them so beautiful and compelling.

Finally, my love goes to my family and friends, without whom I would not have been able to get through this writing process that has taken almost four years. You're the best.

Table of Contents

Preface xiii

Part I. Foundations of Data Systems

1. Reliable, Scalable, and Maintainable Applications 3
Thinking About Data Systems 4
Reliability 6
Hardware Faults 7
Software Errors 8
Human Errors 9
How Important Is Reliability? 10
Scalability 10
Describing Load 11
Describing Performance 13
Approaches for Coping with Load 17
Maintainability 18
Operability: Making Life Easy for Operations 19
Simplicity: Managing Complexity 20
Evolvability: Making Change Easy 21
Summary 22

2. Data Models and Query Languages 27
Relational Model Versus Document Model 28
The Birth of NoSQL 29
The Object-Relational Mismatch 29
Many-to-One and Many-to-Many Relationships 33
Are Document Databases Repeating History? 36

此为试读,需要完整PDF请访问: www.ertongbook.com

Relational Versus Document Databases Today 38
Query Languages for Data 42
Declarative Queries on the Web 44
MapReduce Querying 46
Graph-Like Data Models 49
Property Graphs 50
The Cypher Query Language 52
Graph Queries in SQL 53
Triple-Stores and SPARQL 55
The Foundation: Datalog 60
Summary 63

3. Storage and Retrieval..... 69
Data Structures That Power Your Database 70
Hash Indexes 72
SSTables and LSM-Trees 76
B-Trees 79
Comparing B-Trees and LSM-Trees 83
Other Indexing Structures 85
Transaction Processing or Analytics? 90
Data Warehousing 91
Stars and Snowflakes: Schemas for Analytics 93
Column-Oriented Storage 95
Column Compression 97
Sort Order in Column Storage 99
Writing to Column-Oriented Storage 101
Aggregation: Data Cubes and Materialized Views 101
Summary 103

4. Encoding and Evolution..... 111
Formats for Encoding Data 112
Language-Specific Formats 113
JSON, XML, and Binary Variants 114
Thrift and Protocol Buffers 117
Avro 122
The Merits of Schemas 127
Modes of Dataflow 128
Dataflow Through Databases 129
Dataflow Through Services: REST and RPC 131
Message-Passing Dataflow 136
Summary 139

Part II. Distributed Data

5. Replication..... 151
Leaders and Followers 152
Synchronous Versus Asynchronous Replication 153
Setting Up New Followers 155
Handling Node Outages 156
Implementation of Replication Logs 158
Problems with Replication Lag 161
Reading Your Own Writes 162
Monotonic Reads 164
Consistent Prefix Reads 165
Solutions for Replication Lag 167
Multi-Leader Replication 168
Use Cases for Multi-Leader Replication 168
Handling Write Conflicts 171
Multi-Leader Replication Topologies 175
Leaderless Replication 177
Writing to the Database When a Node Is Down 177
Limitations of Quorum Consistency 181
Sloppy Quorums and Hinted Handoff 183
Detecting Concurrent Writes 184
Summary 192

6. Partitioning..... 199
Partitioning and Replication 200
Partitioning of Key-Value Data 201
Partitioning by Key Range 202
Partitioning by Hash of Key 203
Skewed Workloads and Relieving Hot Spots 205
Partitioning and Secondary Indexes 206
Partitioning Secondary Indexes by Document 206
Partitioning Secondary Indexes by Term 208
Rebalancing Partitions 209
Strategies for Rebalancing 210
Operations: Automatic or Manual Rebalancing 213
Request Routing 214
Parallel Query Execution 216
Summary 216

7. Transactions..... 221
The Slippery Concept of a Transaction 222

The Meaning of ACID 223
Single-Object and Multi-Object Operations 228
Weak Isolation Levels 233
Read Committed 234
Snapshot Isolation and Repeatable Read 237
Preventing Lost Updates 242
Write Skew and Phantoms 246
Serializability 251
Actual Serial Execution 252
Two-Phase Locking (2PL) 257
Serializable Snapshot Isolation (SSI) 261
Summary 266

8. The Trouble with Distributed Systems..... 273
Faults and Partial Failures 274
Cloud Computing and Supercomputing 275
Unreliable Networks 277
Network Faults in Practice 279
Detecting Faults 280
Timeouts and Unbounded Delays 281
Synchronous Versus Asynchronous Networks 284
Unreliable Clocks 287
Monotonic Versus Time-of-Day Clocks 288
Clock Synchronization and Accuracy 289
Relying on Synchronized Clocks 291
Process Pauses 295
Knowledge, Truth, and Lies 300
The Truth Is Defined by the Majority 300
Byzantine Faults 304
System Model and Reality 306
Summary 310

9. Consistency and Consensus..... 321
Consistency Guarantees 322
Linearizability 324
What Makes a System Linearizable? 325
Relying on Linearizability 330
Implementing Linearizable Systems 332
The Cost of Linearizability 335
Ordering Guarantees 339
Ordering and Causality 339
Sequence Number Ordering 343

Total Order Broadcast 348
Distributed Transactions and Consensus 352
Atomic Commit and Two-Phase Commit (2PC) 354
Distributed Transactions in Practice 360
Fault-Tolerant Consensus 364
Membership and Coordination Services 370
Summary 373

Part III. Derived Data

10. Batch Processing 389
Batch Processing with Unix Tools 391
Simple Log Analysis 391
The Unix Philosophy 394
MapReduce and Distributed Filesystems 397
MapReduce Job Execution 399
Reduce-Side Joins and Grouping 403
Map-Side Joins 408
The Output of Batch Workflows 411
Comparing Hadoop to Distributed Databases 414
Beyond MapReduce 419
Materialization of Intermediate State 419
Graphs and Iterative Processing 424
High-Level APIs and Languages 426
Summary 429

11. Stream Processing 439
Transmitting Event Streams 440
Messaging Systems 441
Partitioned Logs 446
Databases and Streams 451
Keeping Systems in Sync 452
Change Data Capture 454
Event Sourcing 457
State, Streams, and Immutability 459
Processing Streams 464
Uses of Stream Processing 465
Reasoning About Time 468
Stream Joins 472
Fault Tolerance 476
Summary 479

12. The Future of Data Systems **489**
Data Integration 490
Combining Specialized Tools by Deriving Data 490
Batch and Stream Processing 494
Unbundling Databases 499
Composing Data Storage Technologies 499
Designing Applications Around Dataflow 504
Observing Derived State 509
Aiming for Correctness 515
The End-to-End Argument for Databases 516
Enforcing Constraints 521
Timeliness and Integrity 524
Trust, but Verify 528
Doing the Right Thing 533
Predictive Analytics 533
Privacy and Tracking 536
Summary 543

Glossary **553**

Index **559**

PART I

Foundations of Data Systems

The first four chapters go through the fundamental ideas that apply to all data systems, whether running on a single machine or distributed across a cluster of machines:

1. Chapter 1 introduces the terminology and approach that we're going to use throughout this book. It examines what we actually mean by words like *reliability*, *scalability*, and *maintainability*, and how we can try to achieve these goals.
2. Chapter 2 compares several different data models and query languages—the most visible distinguishing factor between databases from a developer's point of view. We will see how different models are appropriate to different situations.
3. Chapter 3 turns to the internals of storage engines and looks at how databases lay out data on disk. Different storage engines are optimized for different workloads, and choosing the right one can have a huge effect on performance.
4. Chapter 4 compares various formats for data encoding (serialization) and especially examines how they fare in an environment where application requirements change and schemas need to adapt over time.

Later, Part II will turn to the particular issues of distributed data systems.

此为试读,需要完整PDF请访问: www.ertongbook.com