Broadview
www.broadview.com.cn

[PACKT] open source*
PUBLISHING
community experience distilled

Scala High Performance Programming

高性能Scala

用Scala和函数式编程范式开发高性能软件

[美] Vincent Theron Michael Diamant 著
杨云 廖光明 译

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

[PACKT] open source
PUBLISHING
community experience distilled

Scala High Performance Programming

高性能Scala

用Scala和函数式编程范式开发高性能软件

[美] Vincent Theron Michael Diamant 著
杨云 廖光明 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

Scala 是一种表达能力非常强的语言，能够用非常简洁的代码表达丰富的业务含义。为了在生产上充分发挥 Scala 的能力，除了掌握其简洁的语法外，理解 Scala 在性能上的特点和优化点也是非常重要的事。

本书通过解析一个金融领域高频交易的实际例子，引领读者掌握如何对 Scala 程序（以及一般 JVM 程序）进行基准测试和性能分析，从而找出瓶颈。随后作者介绍了 Scala 语言、Scala 标准库（尤其是集合库）以及 Scalaz 库里解决相应瓶颈的各种技巧。并行计算和分布式架构作为性能调优的重要手段，更是本书的重中之重，作者对 Scala 的并行计算和分布式架构中存在的问题都进行了充分的讨论和讲解，非常值得学习。

本书适合已经具有 Scala 编程基础、能够较好地使用 Scala 风格代码实现业务功能的程序员，作为在性能优化方面的深造阅读资料。

版权贸易合同登记号　图字：01-2016-7237

图书在版编目（CIP）数据

高性能 Scala /（美）文森特·西伦（Vincent Theron），（美）迈克尔·迪亚芒（Michael Diamant）著；杨云，廖光明译. —北京：电子工业出版社，2017.5

书名原文：Scala High Performance Programming

ISBN 978-7-121-31237-3

Ⅰ. ①高… Ⅱ. ①文… ②迈… ③杨… ④廖… Ⅲ.①JAVA 语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2017)第 065972 号

责任编辑：张春雨

印　　刷：三河市良远印务有限公司

装　　订：三河市良远印务有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱　邮编：100036

开　　本：787×980　1/16　印张：15.75　字数：352.8 千字

版　　次：2017 年 5 月第 1 版

印　　次：2017 年 5 月第 1 次印刷

定　　价：69.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

关于作者

Vincent Theron 是一个拥有 9 年工作经验的专业软件工程师。他在 6 年前发现了 Scala 并将之应用于构建高伸缩性、高可靠的应用。他在多个行业设计软件来解决商业上的问题，包括在线博彩、金融交易，以及最近的广告行业。在巴黎东部 Marne-la-Vallée，Vincent 获得了计算机科学及软件工程硕士学位。他和妻子、孩子以及两只毛茸茸的猫一起在波士顿区域生活。

感谢在 Packt 出版社的每一个人，是你们的努力工作才能让这本书面世。感谢柴塔尼亚-耐尔，带着这个出书项目找到我。感谢 Nikhil Borkar，一路上给我提供指引。感谢 Michael Diamant，我的合作作者，我的同事，同时也是朋友，感谢你给这本书带来的知识以及充满灵感的每一天。感谢我的父母，感谢你们的爱和支持，以及你们给我买的第一台电脑。最后，感谢我的妻子，Julie，感谢你一直的鼓励，感谢你给了我一个这么绝妙的儿子。

Michael Diamant 是一个专业的软件工程师，热衷于函数式编程。他在 2009 年开始自己的事业，专注于 Java 和面向对象的编程范式。在 2011 年学习 Scala 之后，他专注于使用 Scala 和函数式编程范式来构建金融交易和广告领域的软件系统。迈克尔毕业于伍斯特理工学院，居住在波士顿区域。

我能在这本书里面分享的知识来源于他人在我一生中给予的支持和教导。我想要特别感谢我的同事文森特，感谢你推动我付出这些努力，感谢所有在一起的时间，让我们可以发展这些书中的想法。我所有的现在和之前的同事都帮助我提高了工程技术，没有你们慷慨地分享你们的所学，我将不可能编写这本书。除了 Vincent，我还想特别提及一些我觉得尤其想感谢的同事：Dave Stevens、Gary Malouf、Eugene Kolnick 和 Johnny Everson。感谢我的父母和兄弟，你们支持我，将我培养成现在的我。我尤其想深深地感谢我的女友 Anna 在整个写书过程中给予我的支持。最后，同样的感谢送给 Packt 出版社，在你们的帮助下我们写了我们的第一本书。

关于审阅者

Nermin Šerifović 是一个从 2009 年开始就热衷 Scala 的人，他从 2011 年开始在专业领域应用 Scala。在大部分工作生涯中，他专注于使用 JVM 技术构建后端平台。最近，作为 Pingup 的研发总监，他正在一个本地化服务预订系统上领导开发工作。

Nermin 是一个哈佛大学延伸教育学院的导师，他在那里与人合作教授 Scala 并行编程的课程，他还在多个学术会议上发表了演讲。

作为一个 Scala 社区的活跃会员，Nermin 组织了波士顿区域的 Scala 爱好者群，同时他也是中东 Scala 座谈基金会一员。他是 *Scala Puzzlers* 一书的合作作者，Scala 解惑网站的合作创建者。

Nermin 拥有康奈尔大学计算机科学硕士学位，他感兴趣的领域包括分布式系统及伴随着的并行，响应式和函数式编程。

前言

Scala 是一个大胆的（audacious）编程语言，在 JVM 平台上，它将面向对象和函数式编程的概念融合到一起。Scala 从一个相对小众的语言成长为了一个开发健壮且可维护的 JVM 应用的首选语言。然而，如果不深入了解这门语言和它提供的高级特性，编写高性能应用依然是一个挑战性的任务。

从 2011 年开始，我们就使用 Scala 来解决复杂的业务上的挑战，这些挑战常常有很高的性能要求。在本书中，我们分享了在这些年中所学到的东西及在编写软件中应用过的技术。我们在书中浏览了这门语言及其生态系统，包括周边工具和使用广泛的库。

我们编写这本书的目标是帮助你理解 Scala 语言提供给你的选择。你有权在书中收集必要的信息，以便在你自己的软件系统中做出更有知识基础的设计和实现决定。我们将不仅给你 Scala 之鱼，带你上路，还将传授渔鱼之技，并给你相关工具让你写出更函数式性能更高的软件。一路上，我们将通过精心设计的业务问题来激发技术讨论，这些业务问题将能让你回想起真实世界的问题。希望通过阅读本书，你能体会到 Scala 的强大能力，并找到合适的工具去编写函数式性能更好的应用。

本书的内容包含什么

第 1 章，通往性能之路，介绍了性能的概念及关于这个主题的一些重要术语。

第 2 章，在 JVM 上衡量性能，详细介绍了 JVM 上可用于衡量和评估性能的工具，包括 JMH 和 Flight Recorder。

第 3 章，释放 Scala 的性能，对多项利用 Scala 语言特性来提升程序性能的技术和模式提供了引导。

第 4 章，探索集合 API，讨论了 Scala 标准库提供的几个集合抽象。在这一章中我们专注于急切执行的集合。

第 5 章，惰性集合和事件溯源，是一个高级技术章节，讨论了两类延迟序列：视图和流。我们还对事件源范式进行了简要的介绍。

第 6 章，Scala 的并发，讨论了编写健壮的并行代码的重要性。我们深入介绍了 Scala 标准库提供的 Future API，并介绍了 Scalaz 库中提出的任务抽象。

第 7 章，高性能架构，作为最后一章，覆盖了前面章节讲过的主题的更深入的相关知识，并探索了使用 CRDT 作为分布式系统的构建块。这一章还探索了在面临高吞吐量时，使用 free monald 实现负载控制策略，以构建响应延时可控的系统。

你需要准备什么

你应该要在你的操作系统上安装好 Java Development Kit 8 版本或更高，以便你可以运行所有的示例代码。本书讨论了 Oracle HotSpot JVM，并演示了在 Oracle JDK 中包含的工具。你应该还需要从 `http://www.scala-sbt.org/download.html` 获取最新版的 sbt（在写作本书的时候，版本为 0.13.11）。

这本书为谁而写

你应该具备关于 Scala 编程语言的基本知识，熟悉一些基本的函数式编程概念，并具有编写产品级 JVM 软件的经验。我们建议刚接触 Scala 和函数式编程的读者在读本书之前花一些时间学习其他的资源，以便可以从书中获取到更多的知识。两个杰出的以 Scala 为中心的资源是 Artima 出版社的《Scala 编程》（*Programming in Scala*）及 Manning 出版社的《Scala 函数式编程》（*Functional Programming in Scala*）。前一个最适用于具有坚实的面向对象 Java 编程知识，同时想要先理解这门语言然后理解函数式编程范式的开发者。后者则聚焦于函数式编程范式而少于讨论某一个具体语言中的概念。

约定

在本书中，你将能找到大量的用于区分不同类型信息的文本样式。下面是一些示例及其所表示的意义。文本中的代码、数据库表名、文件夹名、文件名、文件扩展名、路径名、示例 URL 及用户输入将这样显示：“`-XX:+FlightRecorderOptions` 接受一个名为

settings 的参数，此参数默认值为$JAVA_HOME/jre/lib/jfr/default.jfc.”。

代码块格式如下：

```
def sum(l: List[Int]): Int = l match {
  case Nil => 0
  case x :: xs => x + sum(xs)
}
```

任意的命令行输入和输出格式如下：

```
    sbt 'project chapter2' 'set javaOptions := Seq("-Xmx1G")' 'runMain
highperfscala.benchmarks.ThroughputBenchmark
src/main/resources/historical_data 250000'
```

新的术语和重要的词汇**粗体**显示。在屏幕上面看到的文本，如菜单和对话框，显示为这样的文本样式：让我们从 **Code** 标签组中的 **Overview** 标签开始。

警告或者重要的说明像这样表示。

小建议及技巧像这样表示。

下载示例代码

你可以从 http://www.broadview.com.cn 的“下载专区”，下载所有已购买的博文视点书籍的示例代码文件。

勘误表

虽然我们已经尽力谨慎地确保内容的准确性，但错误仍然存在。如果你发现了书中的错误，包括正文和代码中的错误，请告诉我们，我们会非常感激。这样，你不仅帮助了其他读者，也帮助我们改进后续的出版。如发现任何勘误，可以在博文视点网站相应图书的页面提交勘误信息。一旦你找到的错误被证实，你提交的信息就会被接受，我们的网站也会发布这些勘误信息。你可以随时浏览图书页面，查看已发布的勘误信息。

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），您即可享受以下服务：

- 下载资源：本书所提供的示例代码及资源文件均可在【下载资源】处下载。
- 提交勘误：您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- 与作者交流：在页面下方【读者评论】处留下您的疑问或观点，与作者和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/31237

二维码：

目录

前言 ix

1 高性能之路 1

性能的定义 2

高性能软件 2

硬件资源 3

时延和吞吐率 4

瓶颈 5

性能总结 5

平均数的问题 6

百分位数来救场 8

指标搜集 9

用基准数据（benchmark）来衡量性能 9

通过 Profiling 来定位瓶颈 10

结合基准测试和 profiling 10

案例分析 11

工具链 11

小结 12

2 在 JVM 上度量性能 13

金融领域一瞥 13

意外的市场波动毁掉了利润 16

重现故障 17

吞吐量基准测试 17

时延基准测试 20
定位瓶颈 25
微基准取得大进步 42
小结 49

3 释放 Scala 的性能 51

值类 52
字节码表示 52
性能考虑 54
标记类型——值类的一种替代品 55
专门化 57
字节码表示 58
性能考虑 60
元组 65
字节码表示 65
性能考虑 66
模式匹配 68
字节码表示 68
性能考虑 70
尾递归 75
字节码表示 78
性能考虑 79
Option 数据类型 83
字节码表示 83
性能考虑 84
案例研究——性能更高的 Option 85
小结 89

4 探索集合 API 91

高吞吐量系统 - 改进指令簿 91

理解过去实现上的折中 - list 实现 92
当前的指令簿 - queue 实现 101
通过惰性计算来提升取消操作的性能 104
历史数据分析 114
滞后时序收益率（lagged time series returns） 114
处理多个收益率序列 122
小结 127

5 惰性集合及事件溯源 129

提升用户报表生成速度 129
深入报表生成代码 130
使用视图提速报表生成 133
视图的注意事项 141
打包报表生成结果 145
重新思考报表架构 146
Stream 概览 149
事件变换 152
构建事件源管道 158
马尔可夫流式链 162
流的注意事项 166
小结 169

6 Scala 的并发 171

并行回测（backtesting）策略 171
探索 Future 173
Future 和 crazy ideas 177
Future 使用时的考量 179
提交执行妨碍性能 185
处理阻塞调用和回调 188
ExecutionContext 和阻塞调用 189

用 Promise 转化回调 193
受命进一步提升回测性能 196
介绍 Scalaz Task 197
用 Task 为交易日模拟建模 204
总结回测 209
小结 210

7 高性能架构 211

分布式自动化交易员（Distributed automated traders） 211
分布式架构概述 212
第一次尝试分布式自动化交易系统 212
引入 CRDT 214
CRDT 和自动化交易系统 219
当余额不足时 220
免费交易策略性能提升 222
为交易策略做基准测试 222
无界队列（unbounded queue）的危险 225
应用背压（back pressure） 226
应用负载控制策略 227
Free monad 233
小结 240

1

高性能之路

欢迎加入学习旅程，我们将学会如何用实用的方式用 Scala 编程语言和函数式编程范式来编写高性能、高效率的软件。函数式编程概念，比如纯函数和高阶函数、引用透明和不变性（immutability）是非常理想的工程特性。我们可以用这些特性编写可组合的组件、可维护的软件，以及表达力强且容易理解的代码。但是，尽管有这么多优点，函数式编程经常被（错误地）和低性能、低效的代码联系在一起。本书的目的是告诉你并非如此！本书探索了如何利用函数式编程、Scala 编程语言特性、Scala 标准库和 Scala 生态圈来编写高性能的软件。

Scala 是一种静态强类型的编程语言，它视图优雅地把函数式编程和面向对象编程范式结合起来。最近几年 Scala 日渐流行，因为使用 Scala 你可以既优雅又实用地使用函数式编程范式来开发用于生产环境的软件。Scala 代码编译成字节码，运行在 Java 虚拟机（JVM）上，而 JVM 是一种广为接受的运行平台，可配置，而且提供了优秀的自检（introspect）和调试工具来排查程序错误和性能问题。Scala 的一个附带的好处就是它可以很好地和 Java 互操作，从而使你可以使用已有的 Java 库。尽管 Scala 编译器和 JVM 都在持续地优化，而且现在已经能生成优化良好的字节码，但要达成你的性能目标，仍然需要依靠作为开发者的你的努力。

在一头扎进 Scala 和 JVM 规范前，对于我们所追寻的圣杯——性能——我们先要建立统一的认识。在第 1 章，我们将讨论与编程语言无关的性能概念的基础。我们将提出和解释全书中将要使用的术语和概念。

本章着重讨论以下主题：

- 性能定义
- 性能总结
- 指标搜集

我们还将介绍一个研究案例，一个基于真实问题的虚构应用，这可以帮助我们展示后面章节的技巧和模式。

性能的定义

一个性能词汇表能帮助你限定你手头问题的类型范围，并且能经常指引你找到解决方案。尤其是在时间紧张的情况下，可靠的直觉和有序的策略是你解决性能问题的重要资源。

我们首先要对一个术语建立统一理解：性能。这个术语用于定性地或定量地评估一个软件达成其目标的能力。这个“目标”根据手头的情况，可能差异很大，但是作为专业的软件工程师，这个目标最终必须和业务目的相一致。和业务团队紧密合作来识别出业务上的性能指标是非常重要的。比如对于一个面向终端用户的电商网站，确认并发用户数和可接受的请求响应时间是关键。对于一个金融交易公司来说，交易延迟时间可能是最重要的竞争优势。同时还要注意其他的非功能指标，比如“交易不能丢失”这种行业规范和外部监管要求。这些业务领域约束也会影响你的软件的性能指标。建立明确且被一致认同的领域图景是关键第一步。如果无法明确定义这些约束，那么也无法得出一个有效的解决方案。

需求搜集本身是一个复杂的主题，超出了本书的范畴。如果你想深入这个主题，我们推荐 Gojko Adzic 的两本书：*Impact Mapping: Making a big impact with software products and projects*（http://www.amazon.com/Impa ct-Mapping-software-products-projects-ebook/dp/B009KWDKVA）和 *Fifty Quick Ideas to Improve Your User Stories*（http://www.amazon.com/Fifty-Quick-Ideas-Improve-Stories-ebook/dp/B00OGT2U7 M）。

高性能软件

设计高性能软件是我们作为软件工程师的目标之一。思考这个目标引致一个普遍被问

及的问题：多高的性能算够高？我们用高性能这个术语来描述达到最低可接受阈值的性能表现。我们的目标是达到——如果可能——超出最低可接受阈值。深思一下：如果对于可接受的性能没有统一的标准，显然就不可能产出高性能的软件！这充分说明在开发高性能软件之前，给出明确的性能定义的重要性。

花点时间来思考对你的业务领域来说高性能的定义是什么。你是否曾经为达到你的高性能定义而绞尽脑汁？考虑一下你为解决性能困境采取过的措施。哪些措施是有效的，哪些是无效的？在你阅读本书的过程中，请保持思考这些问题，以便考察哪些技巧能更有效地帮助你达到你的高性能定义。

硬件资源

为了为高性能软件定义指标，我们必须扩展我们的性能词汇表。首先，开始关注你的环境资源。我们用资源这个术语来代表软件运行的所有基础设置。参考下面的资源清单，其中列出了在投入性能调优练习前你需要搜集的全部资源信息：

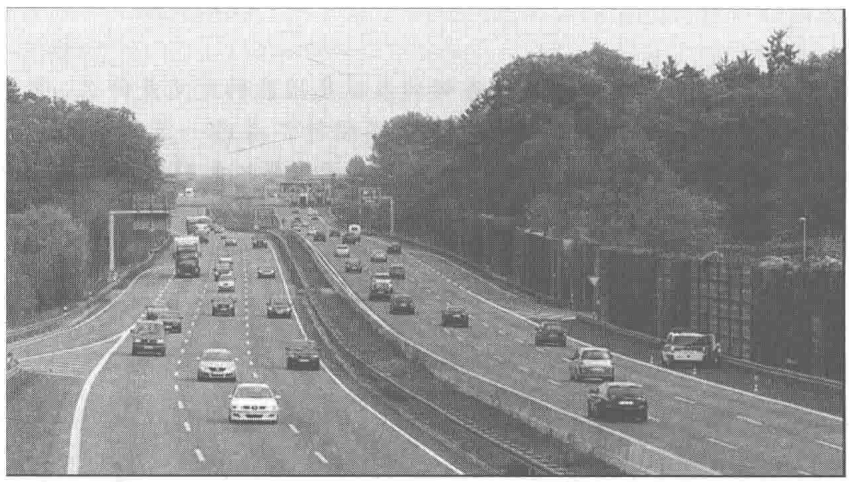
- 硬件类型：物理或虚拟
- CPU：
  核数
  L1、L2 和 L3 缓存大小
  NUMA zone
- 内存（比如 16GB）
- 网络连接速率（比如 1GbE 或 10GbE）
- 操作系统和内核版本
- 内核设置（比如 TCP socket receive 缓存大小）
- JVM 版本

把这些资源清单详细列举可以迫使你思考运行环境的性能极限。

关于内核优化的优质资料包括 Red Hat 性能调优指南（https://goo.gl/gDS5mY）和 Brendan Gregg 的演讲材料和教程（http://www.brendangregg.com/linuxperf.html）。

时延和吞吐率

时延和吞吐率定义了两种不同类型的性能，经常用来建立性能指标。如下图（德国 Autobahn）所示的高速公路图很适合作为比喻来帮助大家建立对这两种性能类型的直观认识：



高速公路的例子能帮助我们理解什么叫时延，什么叫吞吐量。

（图片来自wikimedia，https://en.wikipedia.org/wiki/Highway#/media/File:Blick_auf_A_2_bei_Raststt%C3%A4tte_Lehrter_See_(2009).jpg，图片遵循creative commons许可协议BY-SA 3.0）

时延描述一个被观察的流程完成所花费的时间。以高速公路例子来说，流程是一辆车沿着一条高速公路道行驶。如果高速公路不拥堵，那么车辆可以快速通行。这个流程就被称为低时延。如果高速公路很拥堵，那么通行时间会增加，被称为高时延或缓慢的流程。你能采取的性能优化措施也可以用这个高速公路来比喻。你可以想象把一个昂贵的多项式算法改写成线性算法就好比提高高速路的路面质量或者减少车辆的路面摩擦力。减少摩擦力使车子可以低时延地通过高速路。在实践中，时延性能目标经常用你的业务领域的最大可容忍时延来定义。

吞吐率定义为观察到的流程完成的频率。用高速路类比来说，在单位时间内从 A 点开到 B 点的车辆的数量是高速路的吞吐率。举例来说，如果有 3 条道，每辆车以统一的速率在每条道上行驶，那么吞吐率就是（在观察期内每条道上从 A 点开到 B 点的车数）* 3。归纳推理一下可能会注意到吞吐率和时延之间可能存在强的负相关性（这两个指标是冲突的）。也就是说时延提高时吞吐率下降。事实上，在不少案例中并非绝对如此。请牢记这一点，因为我们将继续扩展我们的性能词汇表来更好地理解为什么会出现这种现象。在实践中，