高等学校计算机教材建设立项项目

高等学校计算机专业规划教材

Java Parallel Programming

Java并行程序设计

张 杨 编著

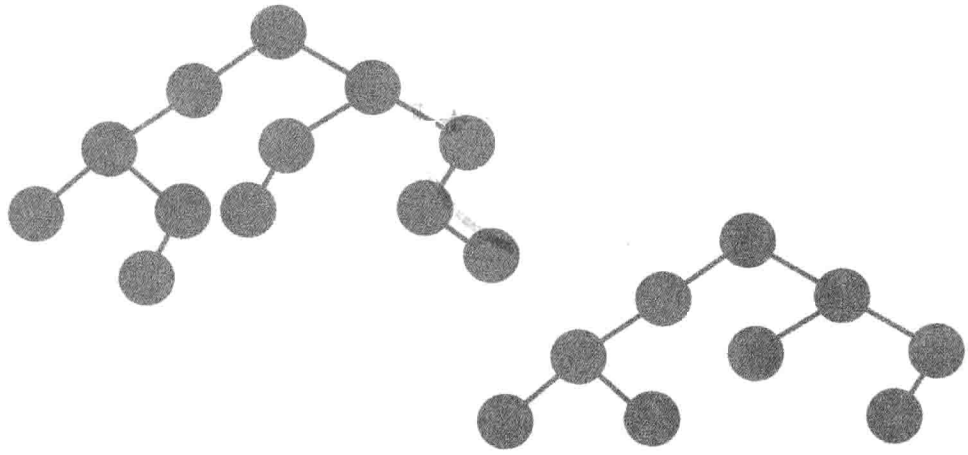
清华大学出版社

高等学校计算机专业规划教材

Java Parallel Programming

Java并行程序设计

张 杨 编著



清华大学出版社
北京

内 容 简 介

本书以 Java 程序设计语言为基础，对并行程序设计的相关概念、基本原理和基本方法进行了介绍，具体涉及线程定义、线程同步、线程障栅、线程间通信、执行器、Fork/Join 框架和自定义的并发类等内容。本书在讲解基本知识的同时，大量使用实例进行演示，演示采用“提出问题、分析问题、代码演示、执行结果、分析结果、相关讨论”的思路，力求做到明白透彻。

本书内容先进、知识结构合理、讲解详尽、例题丰富、深入浅出，适合普通高校、实践和工程类院校学生在学习高性能程序设计时选用，是高等院校学生和 IT 领域在职人员学习 Java 高级编程技术的理想教材和工具书，也可供那些需要高性能计算技术的人员参考。

图书在版编目(CIP)数据

Java 并行程序设计/张杨编著. —北京：清华大学出版社，2015
高等学校计算机专业规划教材
ISBN 978-7-302-39230-9

Ⅰ. ①J…　Ⅱ. ①张…　Ⅲ. ①JAVA 语言－程序设计－教材　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2015)第 024222 号

责任编辑：龙启铭
封面设计：常雪影
责任校对：焦丽丽
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954

印 装 者：北京国马印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：13.75　　**字　　数**：315 千字
版　　次：2015 年 3 月第 1 版　　**印　　次**：2015 年 3 月第1次印刷
印　　数：1～2000
定　　价：29.00 元

产品编号：057785-01

前言

随着多核处理器的普及以及众核处理器设计技术的不断发展，并行程序设计将不会只局限在拥有高性能工作站和服务器的实验室和工业界中，普通用户也可以使用并行程序设计技术设计高性能程序，并行程序设计将成为未来软件开发技术的主流。

并行程序设计继承了传统程序设计中的理论和方法，但对程序设计方法又提出了许多新的要求，突出表现在并行思维能力、并行算法、并行数据分解和任务分解等方面。目前很多程序设计人员对于传统的串行程序设计较为熟悉，对并行程序设计的相关方法还有待于深入学习。

为了帮助更多的人了解和学习并行程序设计的编程思想和方法，编者历时一年半撰写了本书。本书是编者在长期教学和科研实践的基础上编写而成的。在学习本书前，读者应该具备 Java 编程的基本知识。虽然本书介绍的并行程序设计知识只是 Java 世界中之冰山一角，但也覆盖了 JDK 并发库中绝大多数的工具类和接口的使用方法。需要特别说明的是，Java 并发库只是提供了一个工具，并行程序设计过程中最重要的是并行思想和并行程序设计思维习惯的培养。编者希望本书在介绍 Java 并行程序设计相关知识的同时，可以培养学生并行程序设计的思想和思维习惯，引领大家进入到并行程序的世界中，为高性能的程序设计起到抛砖引玉的效果。

支持多线程是 Java 语言的重要特征之一，本书对 Java 多线程编程的相关知识进行了介绍，具体内容涉及线程定义、线程同步、线程障栅、线程间通信、执行器和 Fork/Join 框架等内容，大部分内容在讲解知识的同时都通过实例加以演示，所有例题都由编者亲自编写和测试，实例的演示采用“提出问题、分析问题、代码演示、执行结果、分析结果、相关讨论”的思路，力求做到明白透彻。

本书内容先进、知识结构合理、讲解详尽、例题丰富、深入浅出，是初学者学习并行程序设计的理想教材，适合普通高校、实践和工程类院校学生在学习高性能程序设计时选用，是高等院校学生和 IT 领域在职人员学习 Java 高级编程技术的理想教材和工具书，也可以作为那些需要高性能计算技术人员的自修参考用书。通过本书的学习，读者不但可以了解与并行相关的基本概念，而且能从中学习 Java 多线程的相关知识，使学生在提高 Java 编程能力的同时，还能了解一些并行程序设计的相关思想和方法，为以后的科研和工作打下良好的基础。

全书共11章，其中第1章简要介绍并行程序设计的相关知识；从第2章开始，详细介绍Java线程编程；第3、4和5章讲解Java线程编程中的线程同步、障栅和线程间通信的相关知识；第6、7章介绍了线程执行器和Fork/Join框架；第8章介绍如何自定义并发类；第9章介绍如何使用线程安全的集合操作；第10、11章对多线程程序的性能以及面向方面编程在并行程序设计中的应用进行了介绍。

在本书的编写过程中，得到了河北省重点学科“计算机软件与理论”以及河北省电子信息类本科教学高地重点专业建设项目的支持，得到了国家自然科学基金项目(项目号：61440012)、河北省高等学校青年拔尖人才计划项目(项目号：BJ2014023)和全国高等学校计算机教材建设项目的资助，得到了清华大学出版社的支持，得到了河北科技大学信息科学与工程学院张冬雯院长、王俊社副院长和周万珍副院长的鼎力支持。在写作过程中，杨奎河、张晓明、王井阳、张世民、仇晶、高凯、刘伟、付冬、华宇、王会勇等老师对本书的编写工作提出很多好的意见和建议，国内外众多的Java经典教材、研究成果和相关网站也为本书提供了参考，在此一并表示衷心的感谢。

由于编者学识和水平有限，书中存在不妥之处在所难免，恳请广大读者批评指正。

编　者

2015年2月

目录

第1章 绪 论

Java语言是时下比较流行的一门编程语言，它在面向对象、线程模型、安全性和跨平台等方面提供了良好的支持。在开始学习Java并行程序设计之前，本章介绍一些并行程序设计的基本概念和术语。

1.1 概 述

高性能是计算机系统追求的目标之一，处理器无疑是提高计算机系统性能的核心部件。Intel公司的创始人之一Gordon Moore博士根据处理器的发展规律提出了计算机领域中非常著名的摩尔定律，该定律指出："处理器芯片上集成的晶体管的数量将每18个月翻一番。"按照这个定律，到目前为止处理器上集成的晶体管数量应达到几十万亿个。显然，要达到这个数量，将给处理器的制造商带来更多的困难，这主要因为处理器芯片设计和制造工艺已经达到了极限，而且考虑到功耗和散热等问题，处理器的运算能力已经很难再通过增加晶体管的数量来提高。因此，处理器的制造商不得不采用新的方式来提高计算机的运算能力，IBM等大公司利用并行计算理论的相关知识设计出了多核处理器，希望以此能够继续提高处理器的运算速度。

多核处理器是将两个或多个独立的处理核心集成到一个芯片上，每个内核都有自己的运算单元、控制单元、逻辑单元和中断处理器等。与单核处理器相比，多核处理器能够以较低的频率处理相对多的工作负载。多核处理器的出现，使得处理器处理能力的继续提升成为可能。

目前，工业界中已经生产了许多多核处理器产品，有AMD公司的Opteron和Phenom等系列、Intel公司的Core和Xeon等系列、IBM公司的Power系列以及Oracle公司的UltraSparc系列等。2011年，在美国华盛顿州西雅图召开的Supercomputing大会上，Intel公司正式展示了该公司旗下全新的1Teraflop(每秒钟浮点运算性能万亿次)级别处理器芯片，该处理器芯片拥有多达50个核心。该款处理器的问世，让一颗小小的单芯片处理器相当于配备9680颗英特尔奔腾处理器的超级计算机。

随着多核处理器的普及以及众核处理器设计工艺的不断发展，越来越多的程序将运行在多核平台上。虽然人们对多核处理器寄予厚望，而且多核处理器也确实在一定程度上提升了程序的性能，但是提升的程度与人们当初的期望值还相差甚远，导致这种情况的原因是多方面的，其中最引人注意的无疑是多核处理器硬件体系结构与其上运行的软件架构之间存在巨大的鸿沟，也就是说，软件的运行模型和编程模型与硬件体系结构之间缺

乏有效的对应关系。

当处理器有多个处理核时，传统的串行程序设计在利用这些处理核的能力上就显得有些力不从心了。当没有并行设施辅助时，串行程序只能运行于多核处理器中的一个处理核上，或者在多个处理核上根据操作系统的调度策略进行切换，对多核处理器的利用能力十分有限。因此，为了充分利用多核处理器，需要改变传统的程序编写方式，转向采用并行程序设计方式。

并行程序设计在计算机产生的早期主要集中在专门的实验室和大公司中，并且使用专门的设备进行，参与并行编程的人员都是经过专门培训的人员。多核处理器的出现将改变这一状况，它将使并行程序设计变得更加普及、更加通用，普通的程序员在多核处理器上可以实现程序的并行执行，更多的普通用户也可以参与到并行程序设计的过程中来。

为了提高程序在多核平台上的执行性能，人们提出了三种比较有代表性的解决方法。第一，串行程序自动并行化，即将原有的串行程序通过编译技术或重构技术转换为并行程序，这种方法需要对现有的编译器进行修改和优化，并且需要开发新的重构工具；第二，在串行程序设计语言的基础上增加并发库支持，通过开发相关的并发库，对并行程序设计提供支持；第三，开发全新的并行编程语言，目前，一些大公司已经设计出了并行编程语言，主要有 Oracle 公司的 Fortress、IBM 公司的 X10 和 Cray 公司的 Chapel 等。

Java 从产生之初就可以在语言级别对线程提供支持，它提供了定义和管理多线程的类和方法，使开发多线程程序变得简单、容易和有效。随着 JDK 版本的不断更新，越来越多的并发工具类被加入其中，使用 Java 语言进行编程的能力将越来越强大，在 Java 语言中从事并行编程将变得越来越方便。

1.2 相关概念和术语

1.2.1 并发与并行

并发和并行是两个含义相近的概念。在多核处理器已经普及的年代，虽然人们习惯使用并发或并行表示任务同时运行的含义，对二者不加区分，但是这两个概念还是有一定区别的，有必要首先对并发和并行进行详细说明。

1.2.1.1 并发

并发和操作系统有着密切的关系，并发是操作系统的重要特征之一。在操作系统中允许并发地执行任务，我们可以在听音乐的同时阅读最近发生的新闻，并且还可以发送邮件和撰写文档，这种并发是一种进程(Process)级的并发，是一种程序间的并发执行方式。当然，这种并发也可以发生在程序的内部，我们称其为线程(Thread)级并发。

并发指在某一时间段内，从宏观上看多个程序在同时运行，但在微观上多个程序之间是串行执行的。

图 1-1 给出了程序并发执行的图示，图中共有 4 个线程，分别为线程 1、线程 2、线程 3 和线程 4，从图中可以看出，每个线程的执行都是占用一段时间，线程的执行是分段进行

的，并没有重叠。

图 1-1 并发执行示意图

1.2.1.2 并行

并行指两个或两个以上的任务同时运行，无论从宏观上看，还是从微观上看，任务都是同时运行的。

图 1-2 给出了程序并行执行的图示，从图中可以看出，多个线程之间在某一个时刻（例如 t2 和 t3）是同时在执行的。

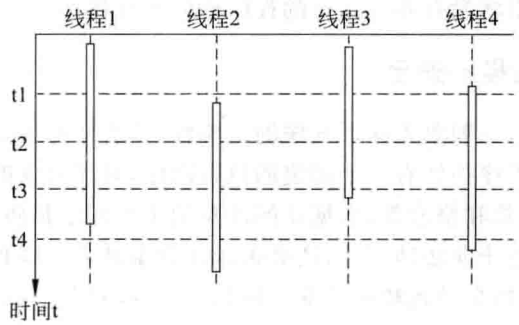


图 1-2 并行执行示意图

1.2.1.3 二者关系

并发和并行是两个既相似又有区别的概念。在英文的表达上，并发的英文是 Concurrency，而并行的英文是 Parallelism，明显不同。二者的区别不仅体现在英文词语的表达上，更体现在微观含义上。

在计算机系统中，可以同时启动多个任务，在任务运行的过程中，究竟是怎样由 CPU 来执行的？如果该计算机只有一个 CPU 或 CPU 处理核，为了使每个任务都有机会运行，CPU 将可以处理任务的时间分成若干片，在每一个时间片中调用一个任务使之运行。在某一段时间内，这种处理方式使得在宏观上看来，多个任务都得到了处理，但从微观上来看，在某一时间点，只有一个任务在运行。总体上多个任务依旧是串行执行的，只不过采用了渐进的方式，让所有的程序都有机会运行一段时间，这种并行方式是一种逻辑意义

上的并行运行。只有当计算机有多个CPU或者一个CPU有多个处理核时，才可以在每个CPU或者在每个处理核上运行一个独立的任务，这些任务不论在微观上还是在宏观上都是同时被处理的，是一种真正意义上的并行运行。

随着多核处理器设计技术的发展，我们所使用的计算机中已有多个处理核，可以同时处理多个任务，这可以看成是一种并行执行的方式。本书没有刻意地区别并发和并行，如果读者使用的是多核处理器，并发和并行的意义是相通的。

1.2.2 串行执行和顺序执行

串行执行是相对于并发/并行执行而言的，并行执行的程序称为并行程序，串行执行的程序称为串行程序。例如有若干个任务，这些任务必须一个接一个执行，这种执行方式是串行执行。如果完成每一个任务需要一定的时间，那么串行执行就是这些任务执行时间的总和。

顺序执行是从程序结构的角度来说明程序的执行。在计算机程序设计语言中，顺序、选择和循环是常用的三种基本结构，其中顺序执行指程序从头到尾地顺序执行，语句的执行将根据书写的先后顺序，程序开头的语句先执行，程序结尾的语句后执行，在未发生异常的情况下每条语句都可以被执行到，而且只被执行一次。

在与并行程序对比时，有些读者喜欢把没有并行执行的程序称为顺序程序，有些读者喜欢称为串行程序，本书将没有并行执行的程序称为串行程序。

1.2.3 线程安全与线程不安全

在程序设计过程中，一般要先保证程序的正确性，其次才是提高程序的性能。在传统的串行执行的程序中，程序往往有一个固定的执行次序，对于数据的访问操作也是有顺序的，例如数据的插入、删除和修改等，数据访问涉及的主要问题是防止非法访问。然而，在多线程程序中，除了要防止非法访问外，还必须保证数据被多个线程操作是安全的。数据被多个线程同时操作可能会导致数据异常。例如，一个线程负责插入输入数据，一个线程负责删除数据，在数据插入之前，如果负责删除数据的线程获得执行机就很有可能导致出错。

一个对象是否是线程安全的，取决于它是否被多个线程访问。当多线程同时访问某个类时，不管线程之间如何交替执行，总能够得到正确的执行结果，则称这个类是线程安全的，否则，则称这个类不是线程安全的。在线程安全的定义中，核心的概念就是正确性，正确性表现为虽然有多个线程在执行某一个类，但就像只有一个线程在执行一样。

要编写线程安全的代码，需要特别注意那些共享的(Shared)和可变的(Mutable)数据或状态的操作。共享意味着变量可以被多个线程所访问，可变意味着变量的值在其生命周期内会发生变化。

线程安全的代码需要采用同步机制来控制对于共享的或可变的变量的访问，特别是多个线程中至少存在一个写操作的情况下。

Java工具集中提供的类有些是线程安全的(例如，类HashTable)，有些则不是(例如，类HashMap)，一般在线程安全的类中都已经封装了必要的同步控制机制，因此不必

进一步采取同步控制措施。这些必要的措施可以包括：

(1) 将线程间共享变量变为线程私有的变量，在线程间不允许共享；

(2) 将状态变量修改为不可变的变量；

(3) 使用同步控制机制。

在设计并行程序时，应尽量考虑到程序运行的环境，设计线程安全的类，这对于增强程序在多核平台上的可移植性是有帮助的。

1.2.4 数据竞争

当只有一个线程访问数据时，数据竞争基本不会发生，只有多个线程同时访问数据时才会发生数据竞争。

数据竞争问题是当有两个或多个同时执行的线程访问同一个内存位置并且至少有一个线程尝试写入数据时而引起的问题。例如，有 A、B 两个线程同时对变量 t 进行操作，如图 1-3 所示。A 线程和 B 线程同时读取了变量 t 的值，A 线程将变量 t 的值增加 10，写回变量 t，B 线程将变量 t 的值增加 20，写回变量 t，显然，后写回的线程会把先写回的线程写入值覆盖。

A 线程	B 线程	t 的值
读取 t=10		10
	读取 t	10
t=t+10，写入		20
	t=t+20，写入	30

图 1-3 两个线程同时对变量 t 操作

为了避免数据竞争，通常需要在程序中加入同步机制，以保证数据访问的正确性。有些同步机制(如锁)可以保证数据在某一时间内只有一个线程访问，有些同步机制(如软件事务性内存)可以让数据由多个线程操作，虽然有多个线程同时访问，但是会保证最早提交的数据有效，而其他数据操作需要回滚。

避免数据竞争的方法有：

(1) 使用同步机制；

(2) 将全局共享数据变为线程私有的数据；

(3) 改变变量的可视范围。

1.2.5 超线程

超线程在字面意义上与线程十分类似，但二者有本质的不同，因此有必要对超线程的概念进行阐述。

超线程是 Intel 公司于 2002 年研发的一种应用于中央处理器(CPU)上的技术，全名为 Hyper-Threading(简称 HT)，中文名为超线程。

尽管通过提高 CPU 的时钟频率和增加缓存容量可以改善 CPU 的性能，但这样的 CPU 性能提高在技术上却存在较大的难度。在实际应用中，很多时候 CPU 的执行单元

都没有被充分使用，这使得 CPU 的性能没有充分发挥。超线程技术是利用特殊的硬件指令，把两个逻辑内核模拟成两个物理芯片，让单个处理器都能使用线程级并行计算，进而兼容多线程操作系统和软件，减少了 CPU 的闲置时间，提高 CPU 的运行速度。

当在一颗 CPU 中同时执行多个程序时，多个程序共同分享一颗 CPU 的资源，理论上像两颗 CPU 一样在同一时间执行两个线程。虽然采用超线程技术能同时执行两个线程，但它并不像两个真正的 CPU 那样，每个 CPU 都具有独立的资源。当两个线程都同时需要某一个资源时，其中一个要暂时停止，并让出资源，直到这些资源闲置后才能继续。因此超线程的性能并不等于两颗 CPU 的性能。

超线程技术起初只应用于 Xeon 处理器中，之后陆续应用在 Pentium 4 中，并将技术主流化。超线程是一个硬件意义上的概念，本书所述的线程属于软件层次的概念。

1.2.6 加速比

在一个多处理机或者多核处理器上运行程序时，很多人常常关心一个问题：并行程序执行的速度比串行程序执行的速度要快多少倍？这里介绍两个著名的定律。

阿姆达尔(Amdahl)定律是计算加速比的著名定律，于 1967 年由阿姆达尔提出，他经过多年的研究，总结加速比的计算公式为：

$$S=\frac{\text{使用单处理器执行程序所用时间}}{\text{使用 p 个处理器(处理核)执行所用的时间}} \tag{1}$$

一个程序通常是由串行执行部分和并行执行部分构成。假定程序中某些部分只能串行执行，如果串行执行部分占整个程序的比率是 f，则并行部分占用的比率为(1−f)，假设并行执行部分无任何其他开销，则用 p 个处理器执行程序所需的时间为：

$$ft_s+\frac{(1-f)t_s}{p} \tag{2}$$

其中，t_s 为使用单处理器执行程序所需的时间。

将公式(2)代入公式(1)，得：

$$S=\frac{t_s}{ft_s+\frac{(1-f)t_s}{p}}=\frac{p}{1+(p-1)f} \tag{3}$$

从公式(3)可以看出，当串行执行部分所占比率 f 为 0 时，S=p，即在并行部分没有任何其他开销的情况下，加速比与处理机数目相同。然而，在实际应用中，串行执行的部分总会占用一定比率，并行执行部分也总会有一些开销，因此在 p 个处理器上执行程序获得的加速比总会小于 p。

偶尔会出现超线性加速比(Superlinear speedup)，即 S>p。通常这是由于使用了优化的并行算法或某一有利于并行程序运行的独特体系结构特性等原因造成的。

Gaustafson 定律是在阿姆达尔定律的基础上提出的一个计算加速比的定律，该定律对加速比的公式描述如下：

$$S=p-f(p-1) \tag{4}$$

公式(4)中各符号的含义同上。

阿姆达尔定律着眼于在固定问题规模情况下可以获得最高加速比，而 Gaustafson 定

律则从另一个角度来看，主张问题的规模应该与可获得的计算资源相匹配。阿姆达尔定律有时用来描述并行化的负面形象，而 Gaustafson 定律则更加正面一些。

1.3 Java 并发方面的特性

随着多核处理器的普及，软件开发人员不得不关注并行编程领域，目前，许多大公司已经提供了对并行程序设计语言的支持，如 Oracle 公司的 Fortress、IBM 公司的 X10 和 Cray 公司的 Chapel 等。在这种背景下，传统的主流程序设计语言 Java 也在不断完善，以适应多核时代给软件开发提出的挑战。

Java 语言从产生之初就支持线程的创建、休眠和终止等操作，并且提供了同步操作。从 JDK 1.5(Java Development Kit)版本开始，由 Doug Lea 领导开发的并发库成为标准库中的一部分，提供了高级的并发工具包 java.util.concurrent，在随后的 JDK 1.6 版本中，增加了一些并行特性，如并行 Collection 框架。2011 年 7 月，Oracle 公司发布了 JDK 1.7，在动态语言类型、垃圾回收、输入输出和并发支持方面都有较大的改进，并且引入了 Fork/Join 框架。2014 年 9 月发布了 JDK 1.8 版本，该版本在 lambda 表达式、标签锁(StampedLocks)和并发计数器等方面又增加了许多新的特性。

在 java.util.concurrent 包中提供了并发开发所需的工具类，这个包包含了几个标准的扩展框架，下面对其中的主要组件进行描述。

- 在早期同步操作的基础上，引入了可重入锁 ReentrantLock 和读写锁 ReadWriteLock 等同步机制，并且提供了 AtomicInteger、AtomicLong 和 AtomicReference 等原子操作。
- 提供了线程执行器框架，该框架包括了接口 Executor 及其子接口 ExecutorService，以及实现了这两个接口的类 ThreadPoolExecutor，该框架可以分离线程的创建和执行操作。执行器起到了维护和管理线程的作用，从而将程序员从繁重的线程管理任务中解放出来。
- 在 java.util.concurrent 包中引入了一个轻量级的 Fork/Join 框架来支持多核环境下的并行程序设计。Fork/Join 框架是并行编程领域中一个经典的框架模型，虽然不能解决所有问题，但在它的适用范围之内，能够轻松地利用多个 CPU 提供的计算资源来协作完成一个复杂的计算任务。通过该框架，程序员可以顺利地过渡到多核时代。
- 在线程障栅操作上，提供了多种障栅操作，如类 CyclicBarrier 和 CountDownLatch 等，并且在 JDK 1.7 版本后实现了类 Phaser，它类似于 CyclicBarrier 和 CountDownLatch，但更灵活。
- 提供了多种线程安全的集合操作，如双端队列、哈希表、跳表等。
- 类 ThreadLocalRandom 提供了线程安全的伪随机数的生成。
- 提供了自定义并发类的功能。