- Java并发开发、互联网项目开发必备书籍
- 线程，线程安全，线程集合类，线程阀，线程池，Fork/Join，线程、线程池在互联网项目开发的应用，线程监控及线程分析，Android中线程应用

Java并发编程从入门到精通

本书示例源代码

张振华 著

清华大学出版社

Java并发编程从入门到精通

张振华 著

清华大学出版社
北京

内 容 简 介

本书作者结合自己 10 多年 Java 并发编程经验，详细介绍了 Java 并发编程的基础概念、工作原理、编程技巧和注意事项，对 Java 高性能高并发编程有极大的参考价值。

本书内容包括并发编程概念，线程，线程安全，线程集合类，线程阀，线程池，Fork/Join，线程、线程池在互联网项目开发的应用，线程监控及线程分析，Android 中线程应用。

本书适合 Java 开发初学者， Java 开发工程师，以及 Java 网络应用优化人员使用，也适合高校相关专业的师生作为课程设计参考使用。

图书在版编目（CIP）数据

Java 并发编程从入门到精通 / 张振华著. -- 北京 ：清华大学出版社，2015
ISBN 978-7-302-40191-9

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

中国版本图书馆 CIP 数据核字（2015）第 101591 号

责任编辑：夏非彼
封面设计：王 翔
责任校对：闫秀华
责任印制：沈 露

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
印 装 者：清华大学印刷厂
经 销：全国新华书店
开 本：190mm×260mm **印 张**：14 **字 数**：358 千字
版 次：2015 年 7 月第 1 版 **印 次**：2015 年 7 月第1 次印刷
印 数：1～3000
定 价：39.00 元

产品编号：064079-01

序　言

古时候，有一个自认为佛学造诣很深的人，听说某个寺庙里有位德高望重的老禅师，便去拜访。老禅师十分恭敬地接待了他，他讲了自己的很多心得，希望老禅师给予指点。

老禅师听后，没有说话，只是为他沏茶。可是在倒水时，明明水已经满了，老禅师还在倒，而不顾茶水都已经溢了出来。最后，这个人终于忍不住说："大师，杯子已经满了。"老禅师这才住手。这个人问老禅师："大师，请你指点。"老禅师说："我已经教你了。"

这个人不明所以，只好回去了。冥思苦想，终于有一天他想明白了：如果自己不把旧茶倒掉，又哪有空间来添续新茶？

空杯心态不仅是一种心境，更是一种做人的境界。其实我们学习和看任何一本书的时候，如果以空杯的心态去看的话，相信收获会更多。功夫巨星李小龙就非常推崇空杯心态，他说："清空你的杯子，方能再行注满，空无以求全。"

最近发现市面上有些书籍要不就是直译过来的，很多不实用，要不就是讲的太玄乎其神了，而此书换一种讲解方式和思路来理解多并发和多线程，让多线程、多并发没有那么玄乎。作者以 10 年的开发经验做总结，希望能帮助读者少走一些弯路，读完这本书让菜鸟变大牛。本书内容安排由浅入深再到应用实践。作者建议大家，不要动不动就 Hadoop，动不动就分布式，将 Java 里面的多并发编程掌握好了，其实就可以解决很多应用问题。

建议大家看此书的时候，结合 JDK 的源码，一起看，每个实例都要运行看看，还要看看咱们工作中，相关的设计是否合理。纸上得来终觉浅，绝知此事要躬行。一定要多加练习才行。

书上有一部分内容是应网友要求编写的，在此表示感谢！也感谢为本书提供精彩书评的朋友。谢谢大家的支持！

本书示例源代码下载地址如下：

http://pan.baidu.com/s/1mgzIbX2

如果下载有问题，请电子邮件联系 booksaga@163.com，邮件主题为"求 JAVA 并发代码"。

著者

2015 年 5 月

推荐语

我在IT软件行业从业已12年。作为“前辈”，衡量一名“程序猿”的技术实力，一般会看你是否具备深度的系统性能调优的能力。云计算的时代，对系统的高性能、高并发要求更高。所以，深入了解和掌握Java的多线程机制原理，非常有用，非常必要。

这本书的所有知识均来自于作者多年的项目实践，倾注了作者多年的心血。讲解的深入浅出，让你掌握起来毫不费力。如果你想成为一名架构师，如果你想成为一名资深的技术大牛，强烈推荐你读一读，你值得拥有！

多家IT公司担任研发总监、技术总监 Justin.Han（韩剑锋）

在进行并发编程开发之前，深入学习并发理论知识非常有必要，比如阅读并发容器的源码。本书通过大量代码实例，讲解并发知识，非常细致。而在实战中并发程序的问题定位也是非常麻烦，相信本书也能给初学者一些参考。

阿里巴巴资深开发工程师，ifeve.com创始人 Kiral（方腾飞）

目　录

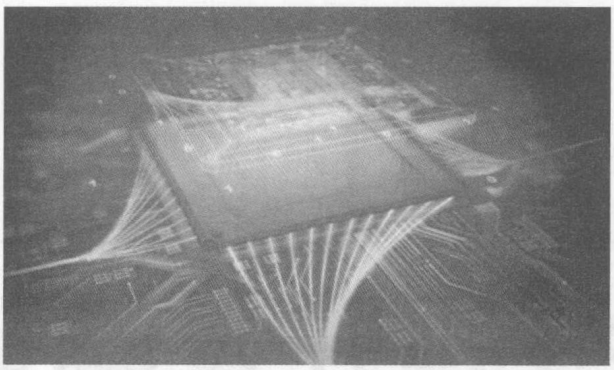
第 1 部分　线程并发基础

第1部分

线程并发基础

第 1 章
概念部分

关注细节求本质，把握机会促发展。



理解互联 Web 开发过程中并发编程相关的必须要知道的一些概念。

1.1 CPU 核心数、线程数

位宽（32 位与 64 位 CPU）32/64 位指的是 CPU 位宽，更大的 CPU 位宽有两个好处：一次能处理更大范围的数据运算和支持更大容量的内存。一般情况下 32 位 CPU 只支持 4GB 以内的内存，更大容量的内存无法在系统识别（服务器级除外）。于是就有了 64 位 CPU，然后就有了 64 位操作系统与软件。

- 多核心：也指单芯片多处理器（Chip Multiprocessors，简称 CMP）。CMP 是由美国斯坦福大学提出的，其思想是将大规模并行处理器中的 SMP（对称多处理器）集成到同一芯片内，各个处理器并行执行不同的进程。这种依靠多个 CPU 同时并行地运行程序是实现超高速计算的一个重要方向，称为并行处理。
- 多线程：Simultaneous Multithreading，简称 SMT。SMT 可通过复制处理器上的结构状态，让同一个处理器上的多个线程同步执行并共享处理器的执行资源，可最大限度地实现宽发射、乱序的超标量处理，提高处理器运算部件的利用率，缓和由于数据相关或 Cache 未命中带来的访问内存延时。当没有多个线程可用时，SMT 处理器几乎和传统的宽发射超标

量处理器一样。SMT 最具吸引力的是只需小规模改变处理器核心的设计，几乎不用增加额外的成本就可以显著地提升效能。多线程技术则可以为高速的运算核心准备更多的待处理数据，减少运算核心的闲置时间。

- 核心数、线程数：目前主流 CPU 有双核、三核和四核，六核也在 2010 年发布。增加核心数目就是为了增加线程数，因为操作系统是通过线程来执行任务的，一般情况下它们是 1:1 对应关系，也就是说四核 CPU 一般拥有四个线程。但 Intel 引入超线程技术后，使核心数与线程数形成 1:2 的关系。

下面我们来看看主流 CPU 的核心数线程数概况。

个人 PC 机，一般是只有一个 CPU：

Intel奔腾双核	Intel酷睿i3	Intel酷睿i5	Intel酷睿i7
双核心 双线程	双核心 四线程	双核心 四线程 四核心 四线程	四核心 八线程 六核心 十二线程 八核心 十六线程

服务器 CPU，与 PC 机的个人电脑的最大不一样是可以有多个 CPU：

IntelXeon 5600	IntelXeon E3	IntelXeon E5	IntelXeon E7
四核心 八线程 六核心 十二线程	双核心 四线程 四核心 八线程	双核心 四线程 四核心 四线程 六核心 十二线程 八核心 十六线程 十二核心 二十四线程	六核心 十二线程 八核心 十六线程 十核心 二十线程

Linux 和 Windows 都有相关的查询以上参数的方法：

Windows 里面查看 CPU 的物理信息，其实就相对简单了，直接在任务管理器里面和设备管理里面就可以看得出来，或者通过第三方工具“鲁大师”等等。

Linux 里面查询 CPU 的物理信息，通过/proc/cpuinfo 这个文件即可查看详情。

（1）查询 CPU 的型号：

```
[root]#  cat /proc/cpuinfo | grep name | cut -f2 -d: | uniq -c
4  Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz
```

可以得出 CPU 的型号是 E5 系列的。

（2）查看 CPU 有几个核几个线程：

```
[root]# grep 'processor' /proc/cpuinfo | sort -u | wc -l
4
[root]# grep 'core id' /proc/cpuinfo | sort -u | wc -l
2
```

可以看得出来有 4 个线程，2 个 CPU 核。

了解你的服务器硬件信息，特别是 CPU 信息，对于做高并发的开发来者来说有益无碍。

1.2 CPU 时间片轮转机制

时间片轮转调度是一种最古老、最简单、最公平且使用最广的算法，又称 RR 调度。每个进程被分配一个时间段，称作它的时间片，即该进程允许运行的时间。

百度百科对 CPU 时间片轮转机制原理解释如下：

如果在时间片结束时进程还在运行，则 CPU 将被剥夺并分配给另一个进程。如果进程在时间片结束前阻塞或结束，则 CPU 当即进行切换。调度程序所要做的就是维护一张就绪进程列表，当进程用完它的时间片后，它被移到队列的末尾。

时间片轮转调度中唯一有趣的一点是时间片的长度。从一个进程切换到另一个进程是需要一定时间的，包括保存和装入寄存器值及内存映像，更新各种表格和队列等。假如进程切（process witch），有时称为上下文切换（context switch），需要 5ms，再假设时间片设为 20ms，则在做完 20ms 有用的工作之后，CPU 将花费 5ms 来进行进程切换。CPU 时间的 20%被浪费在了管理开销上了。

为了提高 CPU 效率，我们可以将时间片设为 5000ms。这时浪费的时间只有 0.1%。但考虑到在一个分时系统中，如果有 10 个交互用户几乎同时按下回车键，将发生什么情况？假设所有其他进程都用足它们的时间片的话，最后一个不幸的进程不得不等待 5s 才获得运行机会。多数用户无法忍受一条简短命令要 5s 才能做出响应。同样的问题在一台支持多道程序的个人计算机上也会发生。

结论可以归结如下：时间片设得太短会导致过多的进程切换，降低了 CPU 效率；而设得太长又可能引起对短的交互请求的响应变差。将时间片设为 100ms 通常是一个比较合理的折衷。

在 CPU 死机的情况下，其实大家不难发现当运行一个程序的时候把 CPU 给弄到了 100%，再不重启电脑的情况下，其实我们还是有机会把它 Kill 掉的，我想也正是因为这种机制的缘故。

1.3 什么是进程和什么是线程

1. 进程是程序运行资源分配的最小单位

进程是操作系统进行资源分配的最小单位，其中资源包括：CPU、内存空间、磁盘 IO 等，同一进程中的多条线程共享该进程中的全部系统资源，而进程和进程之间是相互独立的。进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动，进程是系统进行资源分配和调度的一个独立单位。

进程是程序在计算机上的一次执行活动。当你运行一个程序，你就启动了一个进程。显然，

程序是死的、静态的，进程是活的、动态的。进程可以分为系统进程和用户进程。凡是用于完成操作系统的各种功能的进程就是系统进程，它们就是处于运行状态下的操作系统本身，用户进程就是所有由你启动的进程。

2. 线程是 CPU 调度的最小单位，必须依赖于进程而存在

线程是进程的一个实体，是 CPU 调度和分派的基本单位，它是比进程更小的、能独立运行的基本单位。线程自己基本上不拥有系统资源，只拥有一点在运行中必不可少的资源（如程序计数器，一组寄存器和栈），但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源。

3. 线程无处不在

任何一个程序都必须要创建线程，特别是 Java 不管任何程序都必须启动一个 main 函数的主线程；Java Web 开发里面的定时任务、定时器、JSP 和 Servlet、异步消息处理机制，远程访问接口 RMI 等，任何一个监听事件，onclick 的触发事件等都离不开线程和并发的知识。

1.4 进程与线程比对

以沙箱为例进行对比阐述。一个进程就好比一个沙箱。线程就如同沙箱中的孩子们。孩子们在沙箱子中跑来跑去，并且可能将沙子攘到别的孩子眼中，他们会互相踢打或撕咬。但是，这些沙箱略有不同之处就在于每个沙箱完全由墙壁和顶棚封闭起来，无论箱中的孩子如何狠命地攘沙，他们也不会影响到其他沙箱中的其他孩子。因此，每个进程就像一个被保护起来的沙箱，未经许可，无人可以进出。

如下表所示，进程与线程的对比理解：

	进程	线程
定义	进程是程序运行的一个实体（包括：程序段、相关的数据段、进程控制块 PCB）的运行过程，是系统进行资源分配和调度的一个独立单位	线程是进程运行和执行的最小调度单位
活泼性	不活泼（只是线程的容器）	活泼，随时可以创建和销毁
系统开销	创建、撤消、切换开销大，资源要重新分配和收回	相对于进程仅保存少量寄存器内容，开销小在进程的地址空间执行代码
拥有资产	资源拥有的基本单位	相对于进程来说基本上不拥有资源，但会占用 CPU
地址空间	系统赋予的独立的内存地址空间	线程只由相关堆栈（系统栈或用户栈）寄存器和线程控制表 TCB 组成，寄存器可被用来存储线程内的局部变量
调度	仅是资源分配的基本单位	独立调度、分派的基本单位
安全性	进程之间相对比较独立，彼此不会互相影响	线程共享同一个进程下面的资源，可以相互通信和互相影响

1.5 什么是并行运行

并行运行可以简单做如下理解。

- 第一种情况：程序同时所开启的运行中的线程数，<=CPU 数量*CPU 的核心数量。
- 第二种情况：程序同时所开启的运行中的线程数，<=CPU 数量*CPU 的线程数量。

我们举个例子，如果有条高速公路 A 上面并排有 8 条车道，那么最大的并行车辆就是 8 辆，此条高速公路 A 同时并排行走的车辆小于等于 8 辆的时候，车辆就可以并行运行。CPU 也是这个原理，一个 CPU 相当于一个高速公路 A，核心数或者线程数就相当于并排可以通行的车道；而多个 CPU 就相当于并排有多条高速公路，而每个高速公路并排有多个车道。

1.6 什么是并发运行

当谈论并发的时候一定要加个单位时间，也就是说单位时间内并发量是多少？离开了单位时间其实是没有意义的。

简单来说可以做如下理解：

- 第一种情况：程序同时所开启的运行中的线程数 > CPU 数量*CPU 的核心数量。
- 第二种情况：程序同时所开启的运行中的线程数 > CPU 数量*CPU 的线程数量。

俗话说，一心不能二用，这对计算机也一样，原则上一个 CPU 只能分配给一个进程，以便运行这个进程。我们通常使用的计算机中只有一个 CPU，也就是说只有一颗心，要让它一心多用，同时运行多个进程，就必须使用并发技术。

实现并发技术相当复杂，最容易理解的是“时间片轮转进程调度算法”，它的思想简单介绍如下： 在操作系统的管理下，所有正在运行的进程轮流使用 CPU，每个进程允许占用 CPU 的时间非常短（比如 10ms），这样用户根本感觉不出来 CPU 是在轮流为多个进程服务，就好象所有的进程都在不间断地运行一样。但实际上，因为线程是 CPU 的最小运行单位，在任何一个时间内有且仅有一个线程占有 CPU。如果一台计算机有多个 CPU 或者一个 CPU 有多个核和线程，情况就不同了，如果进程产生的总线程数小于 CPU 的核数，则不同的进程的线程可以分配给不同的 CPU 来运行，这样，各个进程就是真正同时运行的，这便是并行。但如果进程分配的线程数大于 CPU 的核心线程数，则这时候就要使用并发技术。

一个 CPU 就像一条高速公路，如果有条高速公路 A 上面并排有 8 条车道，那么最大的并行车辆就是 8 辆；假设如果一辆车通过一个闸口的时间是 10ms，那么此条高速公路 A 上，1s 通过这个这个闸口的数量就是 1000ms*8/10ms。若果再复杂些，就是每辆车的发动机可能不一样处理的过程可能不一样，不一定都是 10ms，有的慢些，有的快些，这时就用到了“时间片轮转进程调

度算法”，慢的就让他退出，让下一辆先过去，一会又轮到慢的，就是这样交替切换通过闸口关卡。

而对于一个 Web 程序并发来说，1s（或者前端程序展示给用户的最大可容忍时间 3s 等）内的并发量可能计算起来就没这么简单了，影响因素可能会有：程序执行时间，CPU 的时间片轮换时间，程序占用的内存大小，程序占用的宽带大小，请求数据库的时间等等，这里就不多说了。淘宝前台项目工程师蒋江伟曾经给过系统最大并发量的算法，大家没事可以去研究研究。

这里仅仅给一个极限的并发量的算法的例子吧。

当不考虑任何客观因素的情况下，假设一个服务器有 2 个物理 CPU，每个 CPU 有 8 核 16 个线程；那么它的 1s 极限并发量是：1000ms*16(CPU 线程量)*2 个/(CPU 切片轮换时间(假设 10ms)+程序执行时间(假设 10ms))= 1600 个并发。如果前台保证不死机等情况，假设用户可容忍的最大时间是 3s，那么此套程序的最大并发量就是 4800 个。当然了这是一种理想状况，实际上加上其他各种条条框框肯定要比这个少很多。

1.7 什么是吞吐量

吞吐量是指对网络、设备、端口、虚电路或其他设施，单位时间内成功地传送数据的数量（以比特、字节、分组等测量）。

- 网络吞吐量：网络吞吐量是指在某个时刻，在网络中的两个节点之间，提供给网络应用的剩余带宽。即在没有帧丢失的情况下，设备能够接受的最大速率。网络吞吐量可以帮组寻找网络路径中的瓶颈。
- 系统吞吐量：系统吞吐量是指系统在单位时间内所处理的信息量，它以每个时间段所处理的进程数来度量。

1. 网络吞吐量

网络吞吐量是指在某个时刻，在网络中的两个节点之间，提供给网络应用的剩余带宽。即在没有帧丢失的情况下，设备能够接受的最大速率。

（1）吞吐量的大小主要由防火墙内网卡，及程序算法的效率决定，尤其是程序算法，会使防火墙系统进行大量运算，通信量大打折扣。因此，大多数防火墙虽号称 100MB 防火墙，由于其算法依靠软件实现，通信量远远没有达到 100MB，实际只有 10MB~20MB。纯硬件防火墙，由于采用硬件进行运算，因此吞吐量可以达到线性 90MB~95MB，是真正的 100MB 防火墙。

（2）吞吐量和报文转发率是关系防火墙应用的主要指标，一般采用 FDT（Full Duplex Throughput）来衡量，指 64B 数据包的全双工吞吐量，该指标既包括吞吐量指标也涵盖了报文转发率指标。

（3）吞吐量的测试方法：在测试中以一定速率发送一定数量的帧，并计算待测设备传输的帧，

如果发送的帧与接收的帧数量相等，那么就将发送速率提高并重新测试；如果接收帧少于发送帧，则降低发送速率重新测试，直至得出最终结果。吞吐量测试结果以 bit/s 或 B/s 表示。

那这里的吞吐量与带宽有什么区别呢。

吞吐量和带宽是很容易搞混的一个词，两者的单位都是 Mbit/s。先让我们来看两者对应的英语，吞吐量是 throughput，带宽是 Max net bitrate。当我们讨论通信链路的带宽时，一般是指链路上每秒所能传送的比特数。我们可以说以太网的带宽是 10Mbit/s。但是，我们需要区分链路上的可用带宽（带宽）与实际链路中每秒所能传送的比特数（吞吐量）。我们倾向于用“吞吐量”一次来表示一个系统的测试性能。这样，因为实现受各种低效率因素的影响，所以由一段带宽为 10Mbit/s 的链路连接的一对节点可能只能达到 2Mbit/s 的吞吐量。这样就意味着，一个主机上的应用能够以 2Mbit/s 的速度向另外的一个主机发送数据。

2. 系统吞吐量

系统吞吐量是指系统在单位时间内所处理的信息量，它以每小时或每天所处理的进程数来度量。

影响吞吐量因素有：

（1）存储设备的存取速度，即从存储器读出数据或数据写入存储器所需时间。

（2）CPU 性能，即 CPU 的如下 3 个参数：

- 时钟频率。
- 每条指令所花的时钟周期数（即 CPI）。
- 指令条数。

（3）系统结构，如并行处理结构可增大吞吐量。

1.8 高并发编程的意义及其好处和注意事项

由于多核多线程的 CPU 的诞生，多线程、高并发的编程越来越受重视和关注。多线程可以给程序带来如下好处。

（1）充分利用 CPU 的资源

从上面的 CPU 的介绍，可以看的出来，现在市面上没有 CPU 的内核不使用多线程并发机制的，特别是服务器还不止一个 CPU，如果还是使用单线程的技术做思路，明显就 out 了。因为程序的基本调度单元是线程，并且一个线程也只能在一个 CPU 的一个核的一个线程跑，如果你是一个 i3 的 CPU 的话，最差也是双核心 4 线程的运算能力；如果是一个线程的程序的话，那是要浪费 3/4 的 CPU 性能；如果设计一个多线程的程序的话，那它就可以同时在多个 CPU 的多个核的多个线程上跑，可以充分地利用 CPU，减少 CPU 的空闲时间，发挥它的运算能力，提高并发量。