学习攻略 | 如何才能学好并发编程？

2019-02-26 王宝令

讲述：王宝令
时长 12:47　大小 11.71M

并发编程并不是一门相对独立的学科，而是一个综合学科。并发编程相关的概念和技术看上非常零散，相关度也很低，总给你一种这样的感觉：我已经学习很多相关技术了，可还是搞不定并发编程。那如何才能学习好并发编程呢？

其实很简单，只要你能从两个方面突破一下就可以了。一个是“跳出来，看全景”，另一个是“钻进去，看本质”。

跳出来，看全景

我们先说“跳出来”。你应该也知道，学习最忌讳的就是“盲人摸象”，只看到局部，而没有看到全局。所以，你需要从一个个单一的知识和技术中“跳出来”，高屋建瓴地看并发编程。当然，这**首要之事就是你建立起一张全景图**。

不过，并发编程相关的知识和技术还真是错综复杂，时至今日也还没有一张普遍认可的全景图，也许这正是很多人在并发编程方面难以突破的原因吧。好在经过多年摸爬滚打，我自己已经“勾勒”出了一张全景图，不一定科学，但是在某种程度上我想它还是可以指导你学好并发编程的。

在我看来，并发编程领域可以抽象成**三个核心问题：分工、同步和互斥**。

1. 分工

所谓分工，类似于现实中一个组织完成一个项目，项目经理要拆分任务，安排合适的成员去完成。

在并发编程领域，你就是项目经理，线程就是项目组成员。任务分解和分工对于项目成败非常关键，不过在并发领域里，分工更重要，它直接决定了并发程序的性能。在现实世界里，分工是很复杂的，著名数学家华罗庚曾用“烧水泡茶”的例子通俗地讲解了统筹方法（一种安排工作进程的数学方法），“烧水泡茶”这么简单的事情都这么多说道，更何况是并发编程里的工程问题呢。

既然分工很重要又很复杂，那一定有前辈努力尝试解决过，并且也一定有成果。的确，在并发编程领域这方面的成果还是很丰硕的。Java SDK 并发包里的 Executor、Fork/Join、Future 本质上都是一种分工方法。除此之外，并发编程领域还总结了一些设计模式，基本上都是和分工方法相关的，例如生产者 - 消费者、Thread-Per-Message、Worker Thread 模式等都是用来指导你如何分工的。

学习这部分内容，最佳的方式就是和现实世界做对比。例如生产者 - 消费者模式，可以类比一下餐馆里的大厨和服务员，大厨就是生产者，负责做菜，做完放到出菜口，而服务员就是消费者，把做好的菜给你端过来。不过，我们经常会发现，出菜口有时候一下子出了好几个菜，服务员是可以把这一批菜同时端给你的。其实这就是生产者 - 消费者模式的一个优点，生产者一个一个地生产数据，而消费者可以批处理，这样就提高了性能。

2. 同步

分好工之后，就是具体执行了。在项目执行过程中，任务之间是有依赖的，一个任务结束后，依赖它的后续任务就可以开工了，后续工作怎么知道可以开工了呢？这个就是靠沟通协作了，这是一项很重要的工作。

在并发编程领域里的同步，主要指的就是线程间的协作，本质上和现实生活中的协作没区别，不过是**一个线程执行完了一个任务，如何通知执行后续任务的线程开工**而已。

协作一般是和分工相关的。Java SDK 并发包里的 Executor、Fork/Join、Future 本质上都是分工方法，但同时也能解决线程协作的问题。例如，用 Future 可以发起一个异步调用，当主线程通过 get() 方法取结果时，主线程就会等待，当异步执行的结果返回时，get() 方法就自动返回了。主线程和异步线程之间的协作，Future 工具类已经帮我们解决了。除此之外，Java SDK 里提供的 CountDownLatch、CyclicBarrier、Phaser、Exchanger 也都是用来解决线程协作问题的。

不过还有很多场景，是需要你自己来处理线程之间的协作的。

工作中遇到的线程协作问题，基本上都可以描述为这样的一个问题：**当某个条件不满足时，线程需要等待，当某个条件满足时，线程需要被唤醒执行**。例如，在生产者 - 消费者模型里，也有类似的描述，“当队列满时，生产者线程等待，当队列不满时，生产者线程需要被唤醒执行；当队列空时，消费者线程等待，当队列不空时，消费者线程需要被唤醒执行。”

在 Java 并发编程领域，解决协作问题的核心技术是**管程**，上面提到的所有线程协作技术底层都是利用管程解决的。管程是一种解决并发问题的通用模型，除了能解决线程协作问题，还能解决下面我们将要介绍的互斥问题。可以这么说，**管程是解决并发问题的万能钥匙**。

所以说，这部分内容的学习，关键是理解管程模型，学好它就可以解决所有问题。其次是了解 Java SDK 并发包提供的几个线程协作的工具类的应用场景，用好它们可以妥妥地提高你的工作效率。

3. 互斥

分工、同步主要强调的是性能，但并发程序里还有一部分是关于正确性的，用专业术语叫“**线程安全**”。并发程序里，当多个线程同时访问同一个共享变量的时候，结果是不确定的。不确定，则意味着可能正确，也可能错误，事先是不知道的。而导致不确定的主要源头是可见性问题、有序性问题和原子性问题，为了解决这三个问题，Java 语言引入了内存模型，内存模型提供了一系列的规则，利用这些规则，我们可以避免可见性问题、有序性问题，但是还不足以完全解决线程安全问题。解决线程安全问题的核心方案还是互斥。

所谓互斥，指的是同一时刻，只允许一个线程访问共享变量。

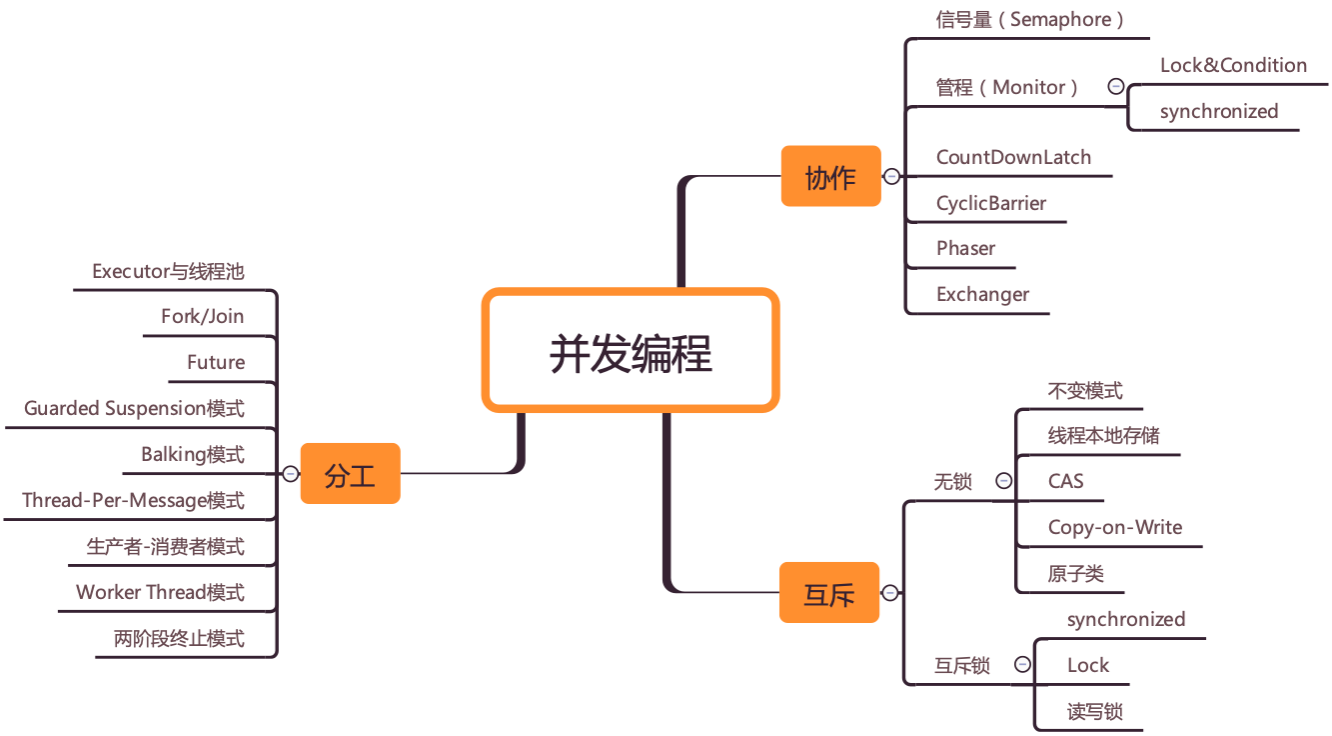
实现互斥的核心技术就是锁，Java 语言里 synchronized、SDK 里的各种 Lock 都能解决互斥问题。虽说锁解决了安全性问题，但同时也带来了性能问题，那如何保证安全性的同时又尽量提高性能呢？可以分场景优化，Java SDK 里提供的 ReadWriteLock、StampedLock 就可以优化读多写少场景下锁的性能。还可以使用无锁的数据结构，例如 Java SDK 里提供的原子类都是基于无锁技术实现的。

除此之外，还有一些其他的方案，原理是不共享变量或者变量只允许读。这方面，Java 提供了 Thread Local 和 final 关键字，还有一种 Copy-on-write 的模式。

使用锁除了要注意性能问题外，还需要注意死锁问题。

这部分内容比较复杂，往往还是跨领域的，例如要理解可见性，就需要了解一些 CPU 和缓存的知识；要理解原子性，就需要理解一些操作系统的知识；很多无锁算法的实现往往也需要理解 CPU 缓存。这部分内容的学习，需要博览群书，在大脑里建立起 CPU、内存、I/O 执行的模拟器。这样遇到问题就能得心应手了。

跳出来，看全景，可以让你的知识成体系，所学知识也融汇贯通起来，由点成线，由线及面，画出自己的知识全景图。



钻进去，看本质

但是光跳出来还不够，还需要下一步，就是在某个问题上钻进去，深入理解，找到本质。

就拿我个人来说，我已经烦透了去讲述或被讲述一堆概念和结论，而不分析这些概念和结论是怎么来的，以及它们是用来解决什么问题的。在大学里，这样的教材很流行，直接导致了芸芸学子成绩很高，但解决问题的能力很差。其实，知其然知其所以然，才算真的学明白了。

我属于理论派，**我认为工程上的解决方案，一定要有理论做基础**。所以在学习并发编程的过程中，我都会探索它背后的理论是什么。比如，当看到 Java SDK 里面的条件变量 Condition 的时候，我会下意识地问，“它是从哪儿来的？是 Java 的特有概念，还是一个通用的编程概念？”当我知道它来自管程的时候，我又会问，“管程被提出的背景和解决的问题是什么？”这样一路探索下来，我发现 Java 语言里的并发技术基本都是有理论基础的，并且这些理论在其他编程语言里也有类似的实现。所以我认为，技术的本质是背后的理论模型。

总结

当初我学习 Java 并发编程的时候，试图上来就看 Java SDK 的并发包，但是很快就放弃了。原因是我觉得东西太多，眼花缭乱的，虽然借助网络上的技术文章，感觉都看懂了，但是很快就又忘了。实际应用的时候大脑也一片空白，根本不知道从哪里下手，有时候好不容易解决了个问题，也不知道这个方案是不是合适的。

我知道根本原因是，我的并发知识还没有成体系。

我想，要让自己的知识成体系，一定要挖掘 Java SDK 并发包背后的设计理念。Java SDK 并发包是并发大师 Doug Lea 设计的，他一定不是随意设计的，一定是深思熟虑的，其背后是 Doug Lea 对并发问题的深刻认识。可惜这个设计的思想目前并没有相关的论文，所以只能自己琢磨了。

分工、同步和互斥的全景图，是我对并发问题的个人总结，不一定正确，但是可以帮助我快速建立解决并发问题的思路，梳理并发编程的知识，加深认识。我将其分享给你，希望对你也有用。

对于某个具体的技术，我建议你探索它背后的理论本质，理论的应用面更宽，一项优秀的理论往往在多个语言中都有体现，在多个不同领域都有应用。所以探求理论本质，既能加深对技术本身的理解，也能拓展知识深度和广度，这是个一举多得的方法。这方面，希望我们一起探讨，共同进步。

欢迎在留言区跟我分享你的经历与想法。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

上一篇　开篇词 | 你为什么需要学习并发编程？

下一篇　01 | 可见性、原子性和有序性问题：并发编程Bug的源头

精选留言 (46)

写留言

Minecraft
2019-02-26

45

并发编程学习 第一天 明天去面试 祝我好运

展开

常江舟
2019-02-26

35

从性能角度讲，我们为了提高执行一定计算机任务的效率，所以IO等待的时候不能让cpu闲着，所以我们把任务拆分交替执行，有了分时操作系统，出现了并发，后来cpu多核了又有了并行计算。这里也就是作者说的[分工]。分工以后我们为了进一步提升效率和更加灵活地达到目的，所以我们要对任务进行组织编排，也就是对线程组织编排。于是线程之间需要通信，于是操作系统提供了一些让进程，线程之间通信的方式。也就是作者说的[...

展开

作者回复: 正解

Handongya...
2019-02-26

12

想给老师提一个建议，就是在开篇用一个问题来引出本篇所要讲述的内容，然后在结尾时的总结之前回答开篇的问题。最后，在总结之后再设计并提出一个问题，让大家来讨论和回答。每一课之后的激烈讨论将是最有意思的，望老师考虑一下，谢谢！

展开

作者回复: 你的建议非常好，我努力向这个方向前进

我会得到
2019-02-26

10

全局思维加单点突破，这种方式屡试不爽。希望令哥沉住气不着急，好好打磨，慢慢更新，搞出精品，打造业界标杆😁

作者回复: 借你吉言

梅云霞
2019-02-26

5

总结

当初我学习 Java 并发编程的时候，试图上来就看 Java SDK 的并发包，但是很快就放弃了。原因是我觉得东西太多，眼花缭乱的，虽然借助网络上的技术文章，感觉都看懂了，但是很快就又忘了。实际应用的时候大脑也一片空白，根本不知道从哪里下手，有时候...

展开

作者回复: 买个专栏啊

王二宝

2019-02-26

4

我和老师的观念是一样的，如果碰到自己一直搞不定的问题时，我的应对方法也是：从两个方面突破。一个是“跳出来，看全景”，另一个是“钻进去，看本质”。

作者回复: 同感

crazypok...

2019-02-26

4

感觉确实如老师所说的，知识不成体系，就像是奶酪，看着是一块，实则满眼孔洞，加油！

展开

作者回复: 这个比喻我是服了

发条橙子 ...

2019-02-26

3

总结 ：

并发编程需要构造出一个全景图 。 只要分为三大点 ： 分工、协作、互斥。

先将一个大的逻辑按不同的工作去分配给不同的线程 ， 这些线程可以同时进行 ，也可...

展开

作者回复: 专栏有专门一期讲管程，每一期都会有相关推荐

邋遢的流浪...
2019-02-26

3

过年的时候看了一遍java并发编程的艺术，感觉有点晕，正好跟着老师的课在深入理解一下

展开

作者回复: 那本书属于高段位的，适合学完这个专栏后再看

minggushen
2019-02-26

2

老师想请教您一个问题，目前公司需要进行分表操作，单表2亿数据，每年的增量也是两亿。有没有什么理论基础支持我分片的片数，以及是否需要分库以及其他注意事项。如果没有的话，老师按照您的经验，应该分成多少个片呢？目前是用的哈希对128取模进行的，分成128个表，是否合适呢。

作者回复: 建议先做个冷热分离吧，如果不能做，建议分库，分片规则很重要，要结合业务，具体问题具体分析。回头我再出个分布式计算的专栏.....

木易走刀口
2019-02-26

2

好东西值得认真学习

展开

作者回复: 感谢杨总支持！

才才
2019-02-26

2

问个问题，分工，同步这俩个概念感觉相似，有本质区别吗

展开

作者回复: 分工主要是拆分任务，要找到瓶颈，设计如何用多线程解决，这个偏设计。同步主要是线程间如何通信，这个偏实现。例如网络io有瓶颈，你可以用一连接一线程来分工，也可以用一组线程监听事件，一组线程处理事件来分工。前一种是不需要同步的，线程间不需要通信。但是后一种需要，因为两组线程要通信

Geek_df121...
2019-02-26

1

18年应届毕业生，从实习生面试到现在，并发使用的并不多，虽然看过一些皮毛，但也是看完就忘。希望可以跟着老师坚持下去，为后面工作和跳槽做准备。

刘付强c...
2019-02-26

1

非常认同“这样一路探索下来，我发现 Java 语言里的并发技术基本都是有理论基础的，并且这些理论在其他编程语言里也有类似的实现。所以我认为，技术的本质是背后的理论模型”，学习一个语言更重要的是理解其内在的“神”而不是外在的“行”，这样才能事半功倍，通吃各种技术门派和各种编程语言。

作者回复: 知己啊！

Weixiao
2019-03-04

老师你好，看一些英文资料，你文中表达的互斥的含义，好像对应这个synchronize这个单词，这个单词被翻译成中文是“同步”，但是英文里说到sync这个概念的时候却在表达你文中所说的“互斥”的概念。你对并发的三个概念划分表示认同，但是中文的叫法，和之前看过的资料有一些不同，望老师指点。

slim
2019-03-03

一边听着歌，一边写Bug。。。

展开

梁中华
2019-03-03

终于有高手站出来总结这块硬知识了，□□

展开

作者回复: 不是高手，只是拿出来大家一起探讨而已

Elon
2019-03-02

Doug Lee有发一些论文比如AQS相关的，也写了一本书Concurrent Programming in Java。可惜比较晦涩没办法通读。

作者回复: 是的，很晦涩。

2019-03-02

我觉得很好，就是一直不能和实际生产整合到一起，在工作中总是无法确定该用怎样的技术，又该从何处使用。

作者回复: 学完万一知道了呢

轻歌赋
2019-03-02

理论方面，老师能解释一下管程的理论吗？
大学课本上面的管程讲的感觉太糙了，一波管程牛逼一吹，给个例子就完了
管程究竟是什么，或者说和别的概念的区别是什么？
系统体系的哪个层次实现其核心代码？
希望老师能帮忙解答一下，谢谢

展开

作者回复: 目录里有介绍，专门一期。