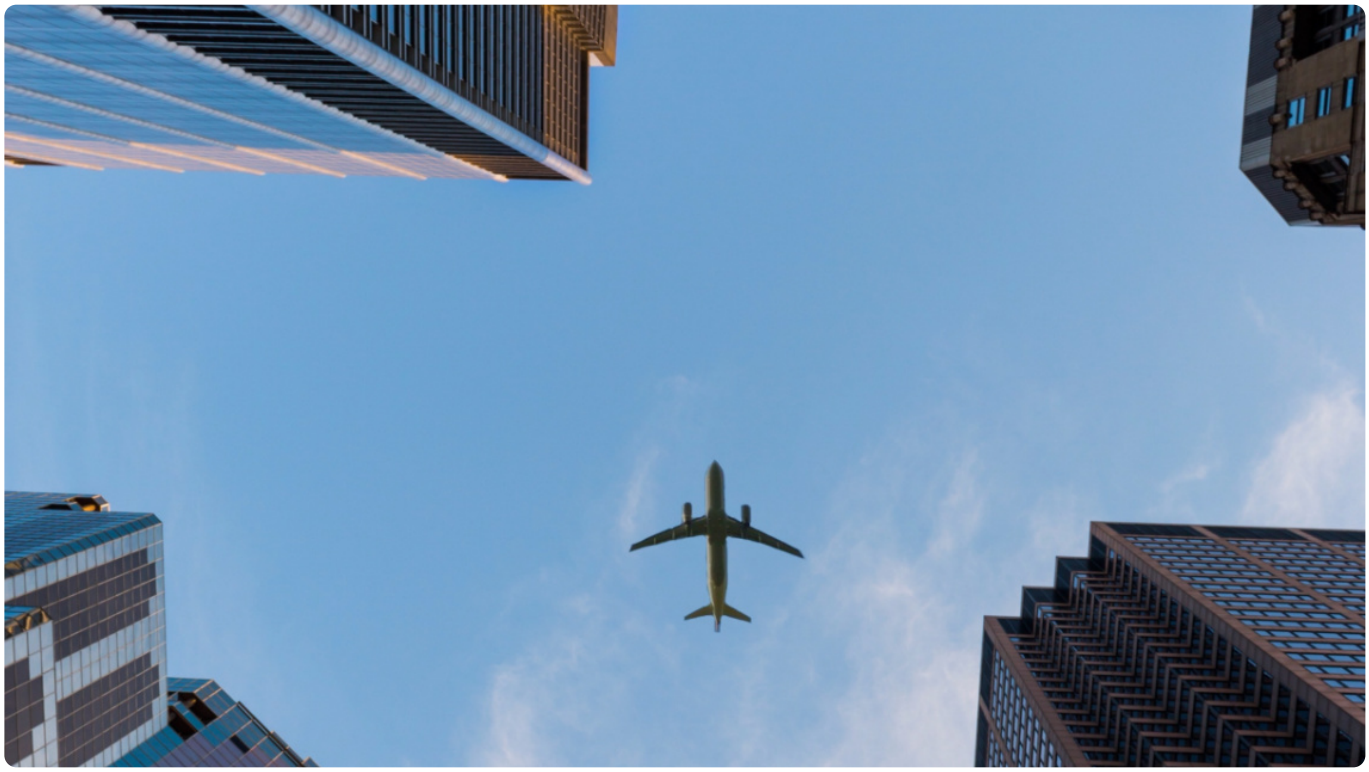
36 | Tomcat I/O和线程池的并发调优

2019-08-03 李号双



讲述：李号双
时长 08:49 大小 8.08M

上一期我们谈到了如何监控 Tomcat 的性能指标，在这个基础上，今天我们接着聊如何对 Tomcat 进行调优。

Tomcat 的调优涉及 I/O 模型和线程池调优、JVM 内存调优以及网络优化等，今天我们来聊聊 I/O 模型和线程池调优，由于 Web 应用程序跑在 Tomcat 的工作线程中，因此 Web 应用对请求的处理时间也直接影响 Tomcat 整体的性能，而 Tomcat 和 Web 应用在运行过程中所用到的资源都来自于操作系统，因此调优需要将服务端看作是一个整体来考虑。

所谓的 I/O 调优指的是选择 NIO、NIO.2 还是 APR，而线程池调优指的是给 Tomcat 的线程池设置合适的参数，使得 Tomcat 能够又快又好地处理请求。

I/O 模型的选择

I/O 调优实际上是连接器类型的选择，一般情况下默认都是 NIO，在绝大多数情况下都是够用的，除非你的 Web 应用用到了 TLS 加密传输，而且对性能要求极高，这个时候可以考虑 APR，因为 APR 通过 OpenSSL 来处理 TLS 握手和加 / 解密。OpenSSL 本身用 C 语言实现，它还对 TLS 通信做了优化，所以性能比 Java 要高。

那你可能会问那什么时候考虑选择 NIO.2？我的建议是如果你的 Tomcat 跑在 Windows 平台上，并且 HTTP 请求的数据量比较大，可以考虑 NIO.2，这是因为 Windows 从操作系统层面实现了真正意义上的异步 I/O，如果传输的数据量比较大，异步 I/O 的效果就能显现出来。

如果你的 Tomcat 跑在 Linux 平台上，建议使用 NIO，这是因为 Linux 内核没有很完善地支持异步 I/O 模型，因此 JVM 并没有采用原生的 Linux 异步 I/O，而是在应用层面通过 epoll 模拟了异步 I/O 模型，只是 Java NIO 的使用者感觉不到而已。因此可以这样理解，在 Linux 平台上，Java NIO 和 Java NIO.2 底层都是通过 epoll 来实现的，但是 Java NIO 更加简单高效。

线程池调优

跟 I/O 模型紧密相关的是线程池，线程池的调优就是设置合理的线程池参数。我们先来看看 Tomcat 线程池中有哪些关键参数：

参数	描述
threadPriority	(int) 线程优先级，默认是5 (Thread.NORM_PRIORITY)
daemon	(boolean) 是否deamon线程，默认为true
namePrefix	(String) 线程前缀
maxThreads	(int) 线程池中的最大线程数，默认是200
minSpareThreads	(int) 最小线程数（线程空闲超过一段时间会被回收），默认是25
maxIdleTime	(int) 线程最大的空闲时间，超过这个时间线程就会回收，直到线程数剩下minSpareThreads个，默认值是一分钟
maxQueueSize	(int) 线程池中任务队列的最大长度，默认是Integer.MAX_VALUE
prestartminSpareThreads	(boolean) 是否在线程池启动时就创建minSpareThreads 个线程，默认为false

这里面最核心的就是如何确定 maxThreads 的值，如果这个参数设置小了，Tomcat 会发生线程饥饿，并且请求的处理会在队列中排队等待，导致响应时间变长；如果 maxThreads 参数值过大，同样也会有问题，因为服务器的 CPU 的核数有限，线程数太多会导致线程在 CPU 上来回切换，耗费大量的切换开销。

那 maxThreads 设置成多少才算是合适呢？为了理解清楚这个问题，我们先来看看什么是利特尔法则（Little’ s Law）。

利特尔法则

> 系统中的请求数 = 请求的到达速率 × 每个请求处理时间

其实这个公式很好理解，我举个我们身边的例子：我们去超市购物结账需要排队，但是你是如何估算一个队列有多长呢？队列中如果每个人都买很多东西，那么结账的时间就越长，队列也会越长；同理，短时间一下有很多人来收银台结账，队列也会变长。因此队列的长度等于新人加入队列的频率乘以平均每个人处理的时间。

计算出了队列的长度，那么我们就创建相应数量的线程来处理请求，这样既能以最快的速度处理完所有请求，同时又没有额外的线程资源闲置和浪费。

假设一个单核服务器在接收请求：

- 如果每秒 10 个请求到达，平均处理一个请求需要 1 秒，那么服务器任何时候都有 10 个请求在处理，即需要 10 个线程。
- 如果每秒 10 个请求到达，平均处理一个请求需要 2 秒，那么服务器在每个时刻都有 20 个请求在处理，因此需要 20 个线程。
- 如果每秒 10000 个请求到达，平均处理一个请求需要 1 秒，那么服务器在每个时刻都有 10000 个请求在处理，因此需要 10000 个线程。

因此可以总结出一个公式：

线程池大小 = 每秒请求数 × 平均请求处理时间

这是理想的情况，也就是说线程一直在忙着干活，没有被阻塞在 I/O 等待上。实际上任务在执行中，线程不可避免会发生阻塞，比如阻塞在 I/O 等待上，等待数据库或者下游服务的数据返回，虽然通过非阻塞 I/O 模型可以减少线程的等待，但是数据在用户空间和内核空间拷贝过程中，线程还是阻塞的。线程一阻塞就会让出 CPU，线程闲置下来，就好像工作人员不可能 24 小时不间断地处理客户的请求，解决办法就是增加工作人员的数量，一个人去休息另一个人再顶上。对应到线程池就是增加线程数量，因此 I/O 密集型应用需要设置更多的线程。

线程 I/O 时间与 CPU 时间

至此我们又得到一个线程池个数的计算公式，假设服务器是单核的：

线程池大小 = （线程 I/O 阻塞时间 + 线程 CPU 时间 ）/ 线程 CPU 时间

其中：线程 I/O 阻塞时间 + 线程 CPU 时间 = 平均请求处理时间

对比一下两个公式，你会发现，**平均请求处理时间**在两个公式里都出现了，这说明请求时间越长，需要更多的线程是毫无疑问的。

不同的是第一个公式是用**每秒请求数**来乘以请求处理时间；而第二个公式用**请求处理时间**来除以**线程 CPU 时间**，请注意 CPU 时间是小于请求处理时间的。

虽然这两个公式是从不同的角度来看待问题的，但都是理想情况，都有一定的前提条件。

1. 请求处理时间越长，需要的线程数越多，但前提是 CPU 核数要足够，如果一个 CPU 来支撑 10000 TPS 并发，创建 10000 个线程，显然不合理，会造成大量线程上下文切换。
2. 请求处理过程中，I/O 等待时间越长，需要的线程数越多，前提是 CUP 时间和 I/O 时间的比率要计算的足够准确。
3. 请求进来的速率越快，需要的线程数越多，前提是 CPU 核数也要跟上。

实际场景下如何确定线程数

那么在实际情况下，线程池的个数如何确定呢？这是一个迭代的过程，先用上面两个公式大概算出理想的线程数，再反复压测调整，从而达到最优。

一般来说，如果系统的 TPS 要求足够大，用第一个公式算出来的线程数往往会比公式二算出来的要大。我建议选取这两个值中间更靠近公式二的值。也就是先设置一个较小的线程数，然后进行压测，当达到系统极限时（错误数增加，或者响应时间大幅增加），再逐步加大线程数，当增加到某个值，再增加线程数也无济于事，甚至 TPS 反而下降，那这个值可以认为是最佳线程数。

线程池中其他的参数，最好就用默认值，能不改就不改，除非在压测的过程发现了瓶颈。如果发现了问题就需要调整，比如 maxQueueSize，如果大量任务来不及处理都堆积在 maxQueueSize 中，会导致内存耗尽，这个时候就需要给 maxQueueSize 设一个限制。当然，这是一个比较极端的情况了。

再比如 minSpareThreads 参数，默认是 25 个线程，如果你发现系统在闲的时候用不到 25 个线程，就可以调小一点；如果系统在大部分时间都比较忙，线程池中的线程总是远远多于 25 个，这个时候你就可以把这个参数调大一点，因为这样线程池就不需要反复地创建和销毁线程了。

本期精华

今天我们学习了 I/O 调优，也就是如何选择连接器的类型，以及在选择过程中有哪些需要注意的地方。

后面还聊到 Tomcat 线程池的各种参数，其中最重要的参数是最大线程数 maxThreads。理论上我们可以通过利特尔法则或者 CPU 时间与 I/O 时间的比率，计算出一个理想值，这个值只具有指导意义，因为它受到各种资源的限制，实际场景中，我们需要在理想值的基础上进行压测，来获得最佳线程数。

课后思考

其实调优很多时候都是在找系统瓶颈，假如有个状况：系统响应比较慢，但 CPU 的用率不高，内存有所增加，通过分析 Heap Dump 发现大量请求堆积在线程池的队列中，请问这种情况下应该怎么办呢？

不知道今天的内容你消化得如何？如果还有疑问，请大胆的在留言区提问，也欢迎你把你的课后思考和心得记录下来，与我和其他同学一起讨论。如果你觉得今天有所收获，欢迎你把它分享给你的朋友。

上一篇　35 | 如何监控Tomcat的性能？

精选留言 (6)

写留言

QQ怪
2019-08-03

猜测请求处理时间过长，应该增大线程池线程数量

展开

作者回复: 这种情况应该怀疑大量线程被阻塞了，应该看看web应用是不是在访问外部数据库或者外部服务遇到了延迟

1　2

nightmare
2019-08-03

查看调用的服务是不是耗时太长

展开

作者回复: 对的

2

门窗小二
2019-08-05

老师！第二种I/O时间与CPU时间这两个指标如何查看？

1

月如钩
2019-08-05

需要排查下外部服务或者数据库连接

展开

-W.LI-
2019-08-03

提高Tomcat最大线程数压榨CPU，优化程序降低响应时间，目测程序阻塞比较多。

许童童
2019-08-03

应该是最大池线程大小设置得过小导致的，具体原因还需要看性能监控和线程池参数配置。