Tomcat并发原理

2021年9月27日 11:33

Tomcat的并发是依赖于多线程来实现的，每一个线程都需要消耗一定的内存用于存储它的运行栈
当线程变多，CPU也需要花费更多的资源进行线程的切换

因为，我们需要根据每一台服务器的物理资源的情况去限制在这台服务器上可以运行的线程数
所以，在Tomcat中会用一个**线程池**的概念来限制服务器上可以运行的线程的数目

在Tomcat中，每一个线程都需要通过io操作，从客户端读取HTTPRequest中的内容，把运行结果放到HTTPResponse中发送回客户端，这样的网络io操作会给服务器带来很大影响

Tomcat的运行模式

1. BIO —— 阻塞IO
2. NIO —— 非阻塞IO
3. APR —— Apache Portable Runtime

这三种运行模式的主要差别在于网络io的处理模式不一样

BIO

BIO是Tomcat7以下默认的运行模式，依赖于传统的java io进行网络io操作，这样一种io是阻塞的，需要等待读写完成才能继续运行

NIO

NIO是基于JDK1.4以后提供的异步IO的API来实现网络io，这样的一种方式在读写网络数据时是不会阻塞的

APR

APR是在操作系统级别来解决异步io的问题，可以比NIO更大幅度的提升服务器的io处理和响应的性能
APR利用Apache的可执行运行库在操作系统上实现高并发，使用时相对麻烦，需要安装Apache的第三方依赖包

BIO的运行模式

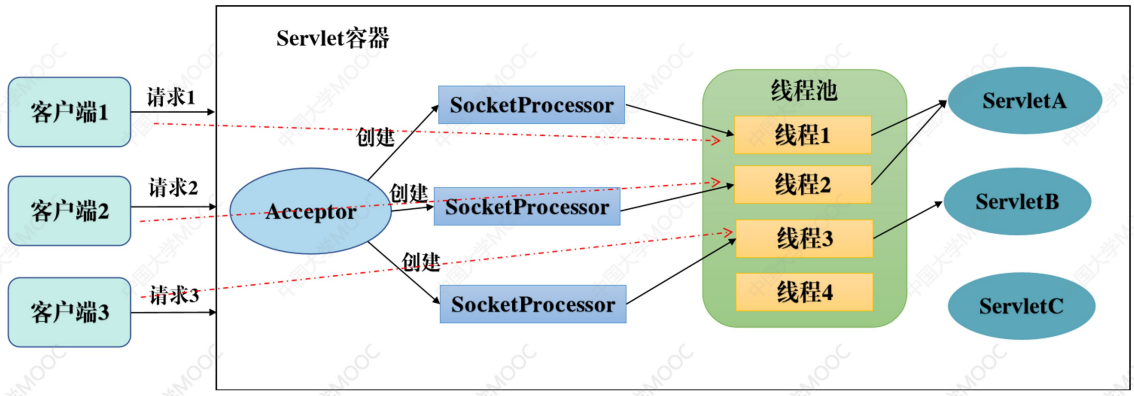
当Tomcat启动时，它会创建并启动一个叫Accpetor的线程，这个线程负责接收客户端的请求。
当它接收到一个客户端的请求后，会创建一个Socket，并将其置于SocketProcessor对象中
然后从线程池中选择一个空闲的线程来执行SocketProcessor对象，在SocketProcessor中，线程会读取Socket来执行网络io，这个过程是阻塞的
（这一段可参照JavaEE/Spring框架/Servlet原理）在读到HTTP请求中的值后，Servlet会创建一个HTTPRequest对象，执行对应service方法。这是简单的处理Http请求的过程

当这个请求正在处理时，如果又收到了另一个请求，Acceptor会创建一个新的Socket，放到新的SocketProcessor对象中，从线程池中选择另一个空闲的线程来应对这个请求

Tomcat不关心请求是针对同一个Servlet还是不同的Servlet，它都会把请求放到不同的线程中去执行，如果容器收到同一个Servlet的多个请求时，这个Servlet的service方法会在多个线程中并发执行

当service方法调用完成后，Servlet容器会把这个线程放回到线程池中。如果当一个请求来到时，线程池中没有空闲线程，这个请求就会被阻塞，放在排队队列中等待，直到线程池中有空闲线程
我们可以为Servlet容器配置最大排队数，如果超出最大排队数，Servlet容器就会拒绝相应的http请求

• 线程模型 - BIO



NIO运行模式

NIO同样由Acceptor去接收请求，收到请求后，会建立Socket，但它**不是把Socket放到SocketProcessor中**，而是创建一个NioChannel，把Socket以及创建的Buffer放到NIOChannel对象中

NioChannel会从Socket中读取数据，放到Buffer中。数据读取完成后，NioChannel会发事件去通知Servlet容器的线程去读取数据

Servlet如何知道NioChannel读取数据完成?

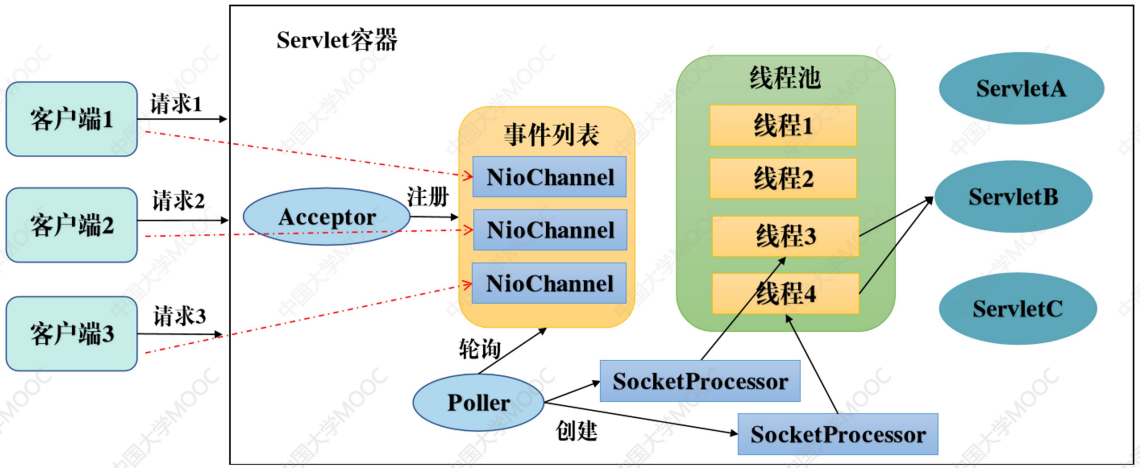
Accpetor会把NioChannel登记到一个事件列表中，由一个Poller线程去负责**轮询**事件列表，去看哪一个NioChannel已经接收完成了数据。

如果某一个NioChannel已经完成了数据的接收，那么就创建一个SocketProcessor，把Socket放到SocketProcessor中。之后的过程同BIO

不同的是，当线程读取数据时，**数据已经存在Buffer中**了，所以读取速度会稍快一些

与BIO相比，如果网络性能是系统瓶颈，那么用NIO可以大幅度提高系统性能，可以支持更多的请求

• 线程模型 - NIO



Tomcat并发的特点

Tomcat的并发处理是依赖于多线程来完成的，主要思想是无论有多少个请求过来，对CPU和内存的消耗应该是相对恒定的

因此，Tomcat主要通过两个机制来固定对CPU和内存的消耗

1. 单实例Servlet对象

   无论有多少请求过来，在Serlvet容器中间，每一个Servlet类只有一个实例对象，不会随着请求的增多而增多

2. 基于线程池的多线程

   通过线程池来固定线程的数目，如果线程超过这个数目，就将其置于等待队列

3. 一个问题：Servlet对象线程不安全

   因为是单实例Servlet对象，针对同一个Servlet的多个请求是共用同一个Servlet对象，这个Servlet对象的属性是会被多个线程共享的

   所以我们不会在Servlet的对象上去写任何属性

   如果要写属性，需要加锁，而加锁又会限制进程的并发性