胖虎的个人博客

别闻，闻就-1s，一分钟有59s就van♂s了

Java NIO学习与记录（六）： NIO线程模型

NIO线程模型

上一篇说的是基于操作系统的IO处理模型，那么这一篇来介绍下服务器端基于IO模型和自身线程的处理方式。

一、传统阻塞IO模型下的线程处理模式

这种处理模型是基于阻塞IO进行的，上一篇讲过，阻塞IO会阻塞每一个IO操作，直到事件就绪，下面来看下阻塞IO下的服务端线程模型：

公告

GitHub

昵称： 胖虎1993

园龄： 10个月

粉丝： 6

关注： 2

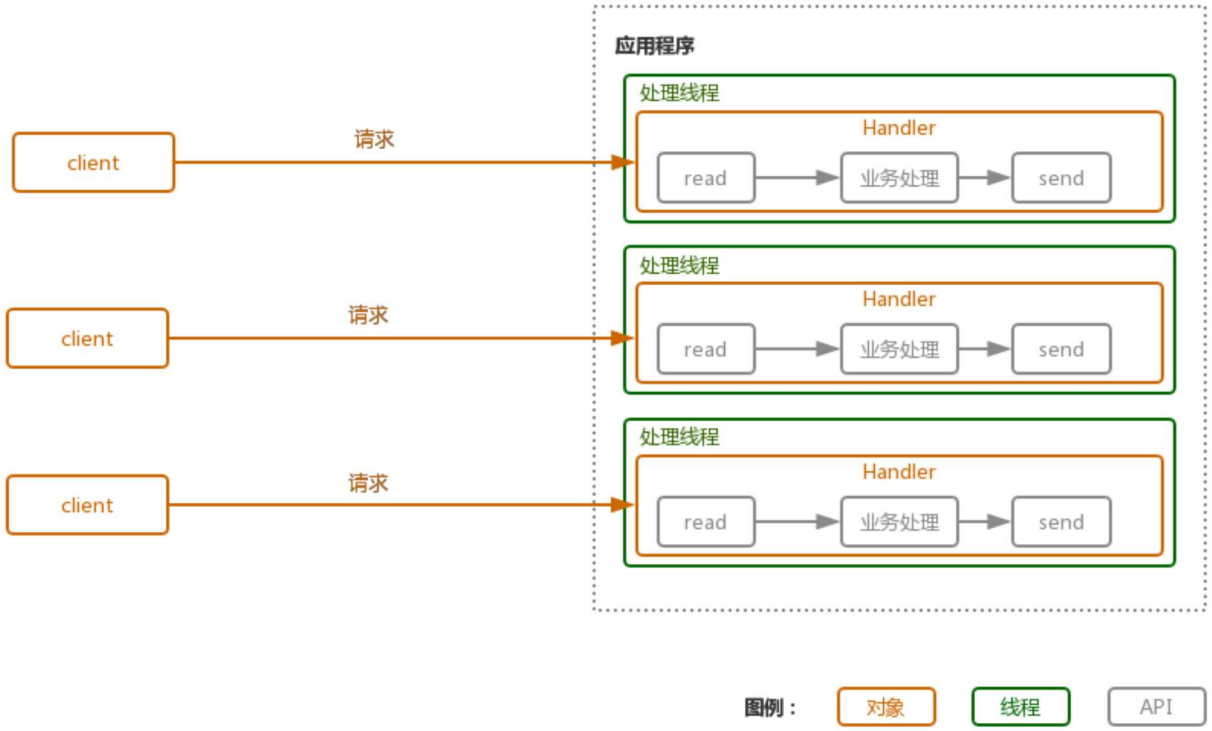


图1

如上图所示，该线程模型基于阻塞IO模型实现，针对每个请求都需要抽出来一个线程进行处理读入数据、业务处理数据、返回响应结果给客户端，这个过程中读、写操作均会阻塞，且跟业务处理串行执行，该模式下，并发量过大时会大量创建线程，发生的大量上下文切换，从而导致CPU资源占用过大，当连接建立后，若当前线程暂无可读数据，则线程会一直阻塞在读操作上，造成线程资源浪费，即便使用线程池进行优化，虽然避免了大量创建线程，但也会出现线程资源浪费的问题，高并发下可能会造成排队、响应不及时的问题。

具体BIO服务器的实现参考：SocketChannel与BIO服务器

二、基于IO多路复用模型下的Reactor线程模型

利用操作系统IO多路复用模型实现，Java对其调用进行了封装（select等），这里先不探讨怎么利用java的api去调用，先来看看它的基本流程是怎样的，Reactor模式下的县线程模型又会根据线程数量、线程池数量的不同，细分了三种线程模型。

2.1：单Reactor单线程模型

这是最简单的Reactor模型，整个过程中的事件处理全部发生在一个线程里：

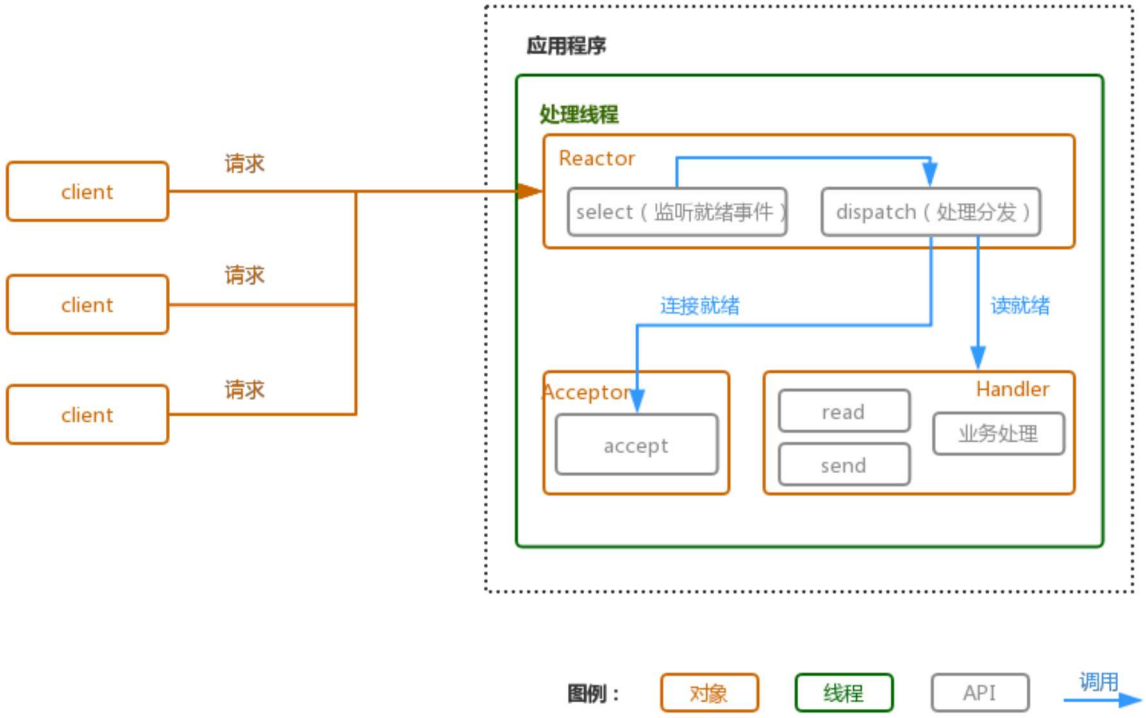


图2

上图示意就是个简单的IO多路复用单Reactor单线程处理模型，流程如下：

①Reactor对象通过select监听客户端的请求事件，收到事件消息后通过dispatch进行任务分发。

②如果是建连请求，则交由Acceptor对象处理连接请求，然后创建一个Handler对象继续完成后续处理

③若不是建连请求，则dispatch会调用对应连接的Handler进行处理，Handle负责完成连接成功后的后续处理（读操作、写操作、业务处理等）

我的标签

我的标签

连接池(14) druid(13) java(10) NIO(9) Socket(8) ThreadLocal(5) 多线程(3) 链路追踪(2) 设计模式(2) spring(2) 更多

随笔分类

grpc
InfluxDB(1)
java(12)
mysql(2)
Netty
NIO(8)
redis(2)
spring(2)
多线程(18)

此模型很简单，易于理解，但是存在一定的问题，比如单线处理程模型下，无法发挥多核CPU的性能，如果Handler上的业务处理很慢，则意味着整个程序无法处理其他连接事件，造成性能问题。

适用于业务处理快速、客户端连接较少的情况。

2.2：单Reactor多线程模型

相较于上面的模型，对业务处理模块进行了异步处理，流程图如下：

缓存设计(1)
监控相关(3)
日常(5)
设计模式(2)
数据结构(3)
网络编程(8)
杂记(6)

随笔档案

2019年10月(1)
2019年9月(12)
2019年8月(1)
2019年7月(1)
2019年6月(1)
2019年5月(1)
2019年4月(6)
2019年3月(13)
2019年2月(8)

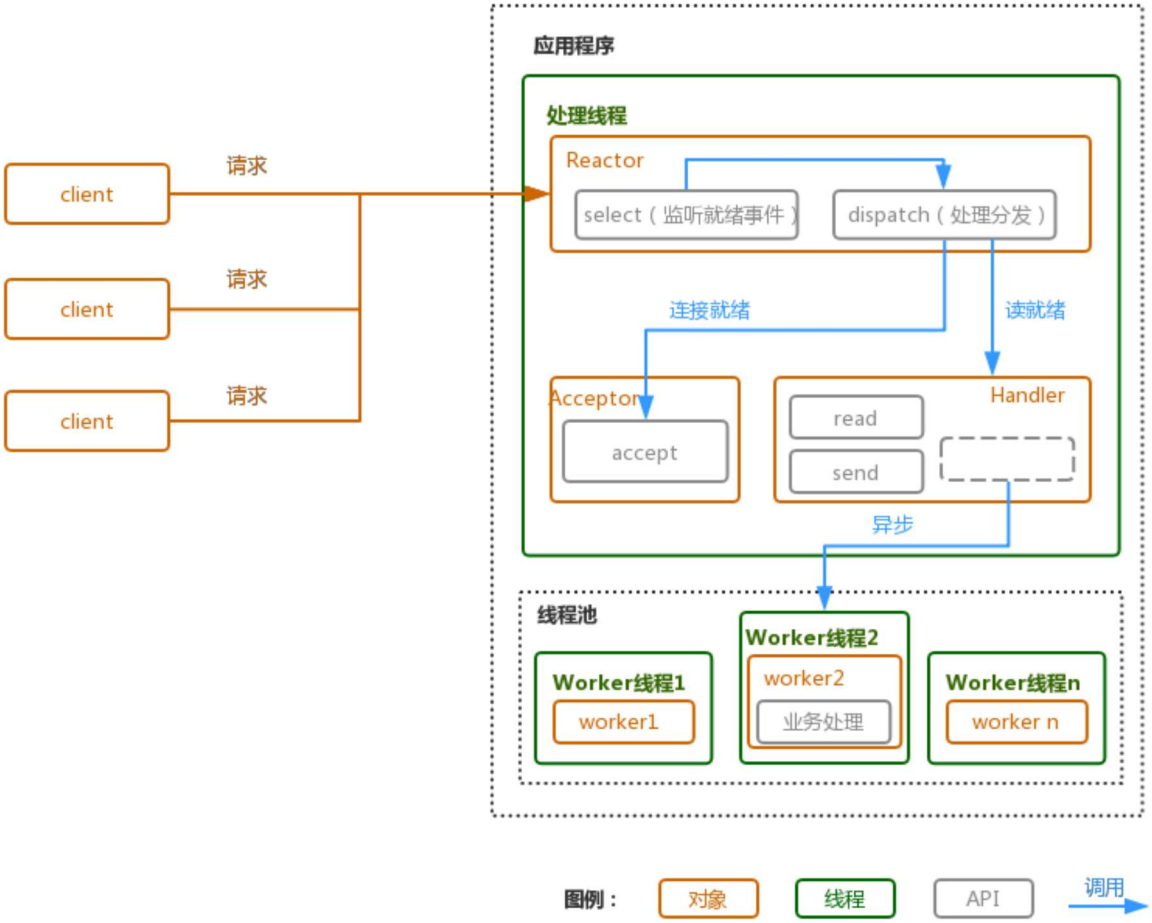


图3

上图示意属于单Reactor多线程处理模型，流程如下：

①Reactor对象通过select监听客户端的请求事件，收到事件消息后通过dispatch进行任务分发。

②如果是建连请求，则交由Acceptor对象处理连接请求，然后创建一个Handler对象继续完成后续处理

2019年1月(4)
2018年12月(2)

相册
PC壁纸(21)
春夏秋冬(18)
手机壁纸(8)
素材(9)
阴阳师壁纸(6)

快速通道
哔哩哔哩
AcFun
网易云音乐
GitHub
Stack Overflow
OsChina

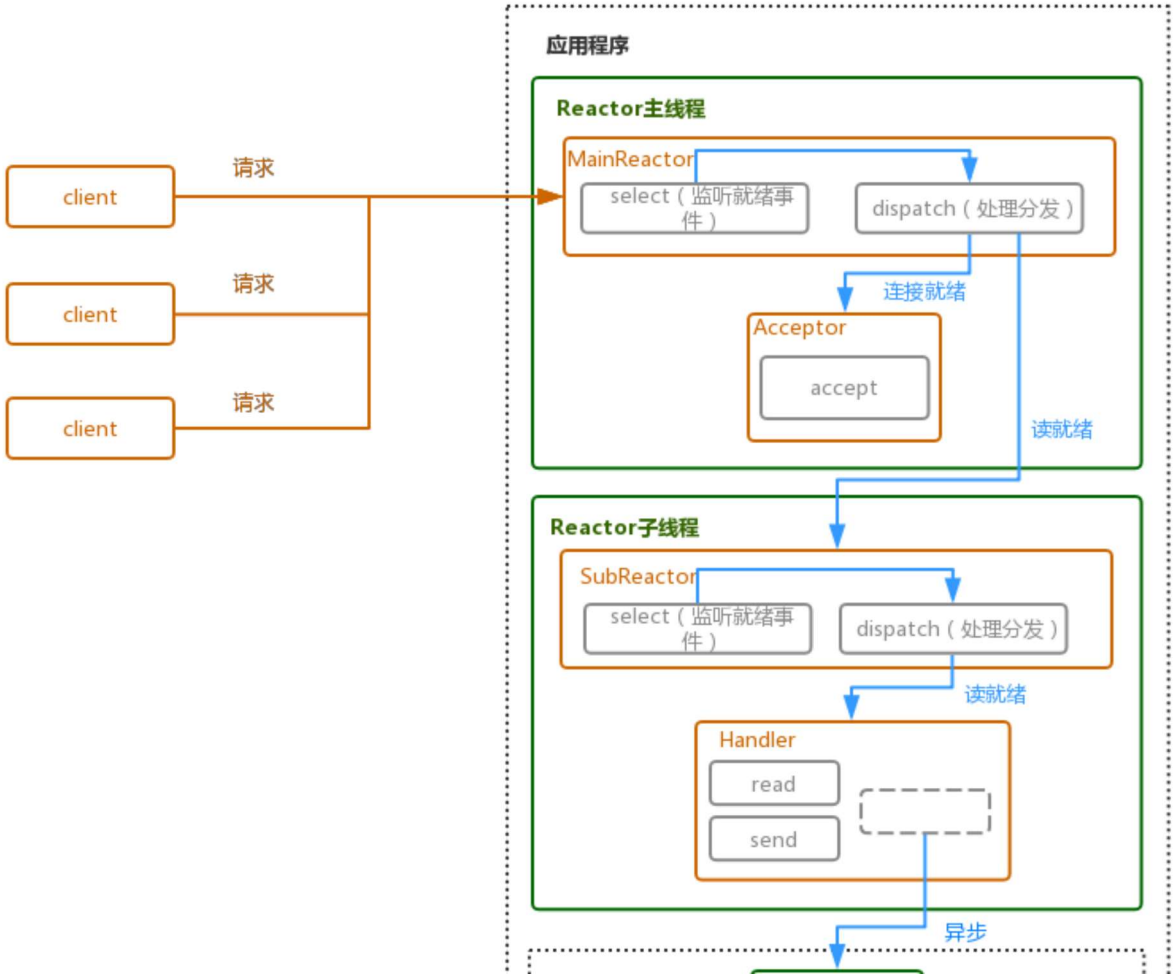
阅读排行榜

③若不是建连请求，则dispatch会调用对应连接的Handler进行处理，Handle负责完成连接成功后的读操作，读出来数据后的业务处理部分交由线程池异步处理，业务处理完成后发送给Handler处理完成的消息，然后再由Handler发送处理响应信息给对应的Client。

本模型充分利用了多核CPU的处理能力，降低了由业务处理引起的性能问题，Reactor线程仅负责接收连接、读写操作。但是Reactor除了负责连接处理外仍然负责读写操作，大量的请求下仍然可能仍然存在性能问题。

2.3：主从Reactor多线程模型

这个模型中将会独立出另一个Reactor对象来处理非连接处理的其他处理，命名为从Reactor（SubReactor），流程图如下：



1. ThreadLocal系列（三）-TransmittableThreadLocal的使用及原理解析(842)

2. [温故]图解java多线程设计模式（一）(624)

3. 【初探】java性能火焰图的生成(447)

4. 链路追踪（一）-分布式链路追踪系统的介绍(397)

5. Java NIO学习与记录（六）： NIO线程模型(284)

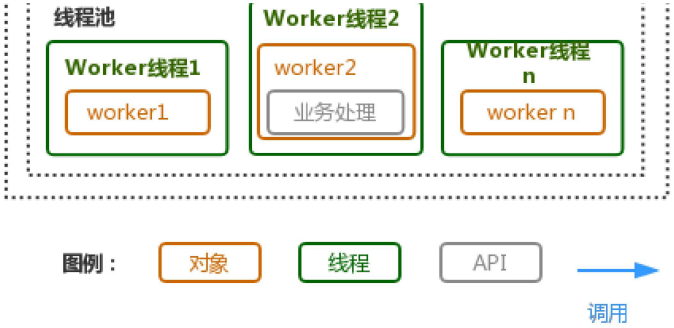


图4

上图示意属于主从Reactor多线程处理模型，流程如下：

①主Reactor对象（MainReactor）通过select监听客户端的连接事件，收到连接事件后交由Acceptor处理。

②Acceptor处理完成后，MainReactor将此连接分配给SubReactor处理，SubReactor将此连接加入连接队列进行事件监听并建立Handler进行后续的各种操作，同上面的模型一致，SubReactor会监听新的事件，如果有新的事件发生，则调用Handler进行相应的处理。

③Handler读出来数据后的业务处理部分交由线程池异步处理，业务处理完成后发送给Handler处理完成的消息，然后再由Handler发送处理响应信息给对应的Client。

该模型存在两个线程分别处理Reactor事件，主线程只负责处理连接事件，子线程只负责处理读写事件，这样主线程可以处理更多的连接，而不用关心子线程里的读写处理是否会影响到自己。目前这种模型被广泛使用在各种项目中（如Netty、Memcached等）。

以上的线程模型都是基于同步IO，异步IO这里不作说明，目前大部分项目都采用IO多路复用（同步非阻塞）的模式进行（该模式下又分成了上述3种线程处理模型）。

下一篇将会针对IO多路复用下的三种线程模型，介绍下Selector，以及利用Selector来写一下具体的实现代码。

分类：NIO, 多线程, 网络编程

标签：NIO, Socket