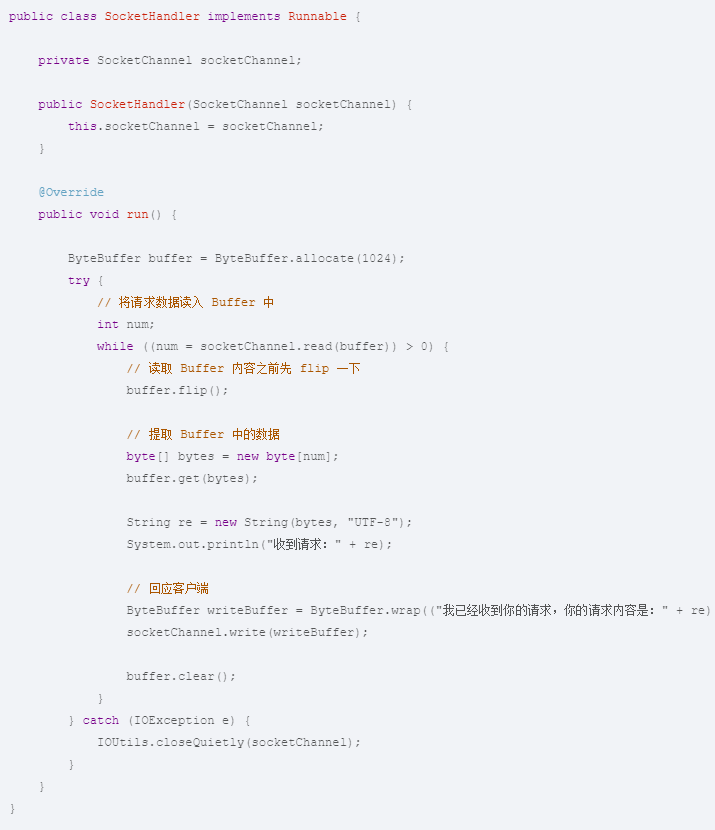
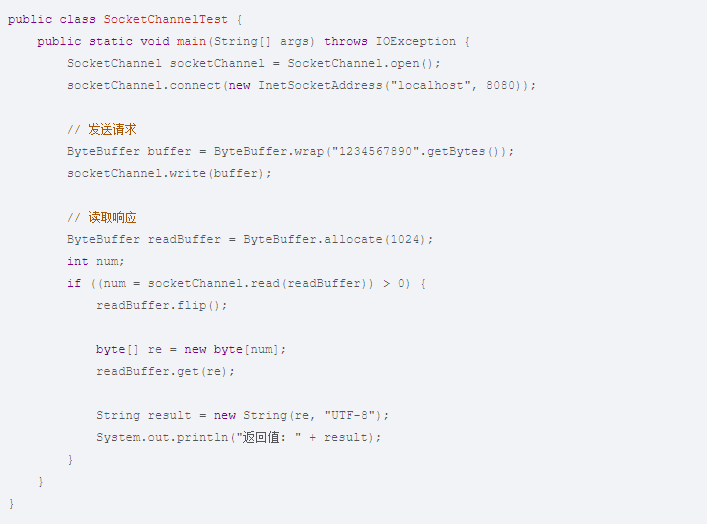
## BIO

直接贴代码：





客户端：



BIO处理逻辑：每来一个连接就交给一个线程全程处理。

瓶颈：

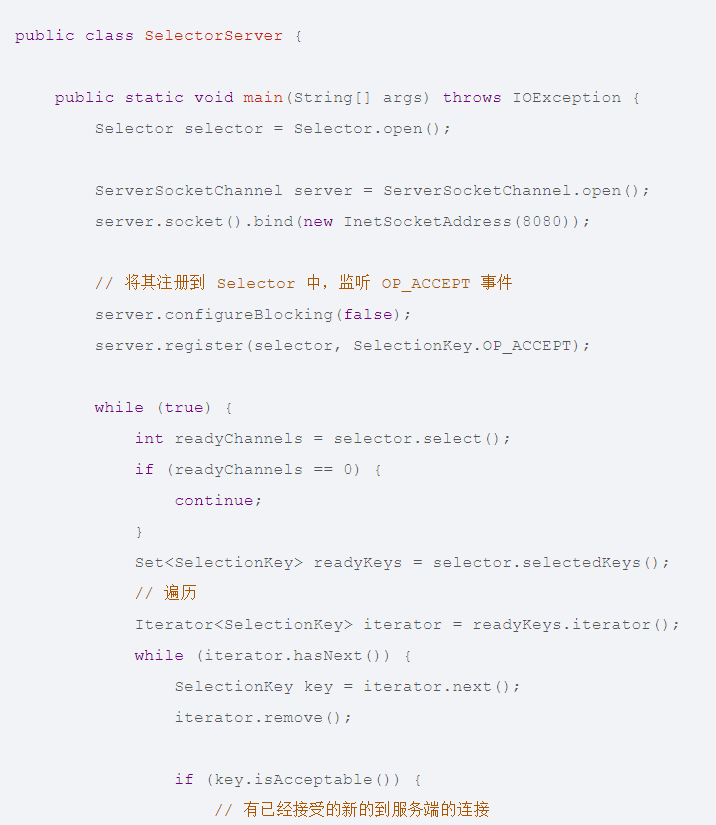
1. 每一个连接都新分配一个线程去处理显然不合适。当请求上万甚至几十万的时候，内存资源会被迅速耗光，并且线程切换的开销也是一个不可忽略的。
2. BIO的阻塞会更多。首先accept()是一个阻塞操作，当新连接过来的时候，read()需要等待通道可读，才能执行读操做，写操作同理。
3. 读写的阻塞操作原因：应用层上的读写操作是数据在用户空间缓存和内核缓存之间相互copy，当写操作的时候，数据从用户空间的缓存copy到内核缓存，需要阻塞到所有数据copy完毕，期间包括等待操作系统的网络IO，将数据从本地传输到远程计算机。当读操作的时候，数据从内核缓存copy到用户空间缓存，直到返回-1，期间包括操作系统接收远程网络IO过来的数据。

解决办法：

1. 可以看到，读写操作有一部分时间是阻塞着的，只是干等着本机将数据准备好。如果将干等着的时间去做其他事，岂不是可以提高效率。

# NIO

NIO其实是一种同步非阻塞IO。

代码改进：当监听到一个事件key.isAcceptable()、key.isReadable()、key.isWritable()应该新开启一个线程做剩下的IO操作。Tomcat的NIO使用了这种方式。

NIO优势：

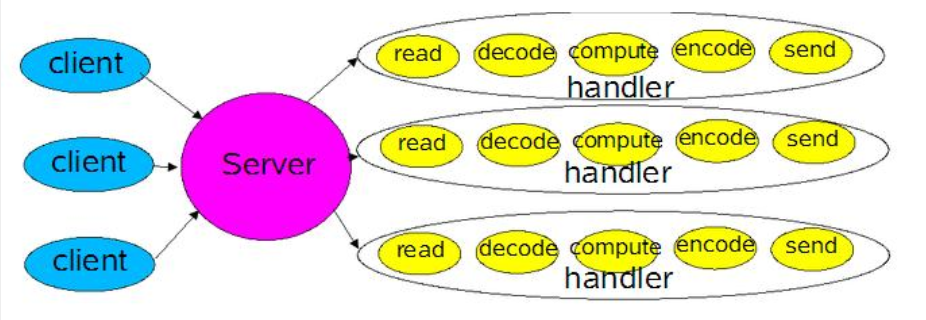
1. 通过监听到的不同事件，执行对应的操作。这样就节省了读写的“干等”时间。
2. NIO的阻塞操作是selector.select(),就是说一个selector可以同时管理N个通道，取代了原本的N个accept()阻塞操作。

# NIO及Reactor模式

IO操作可以看出是三种操作的集合：连接、读、写。传统IO模型中，将这三种操作交给同一个线程处理，因此带来了较大的阻塞开销。在NIO中，selector将这三种操作区分开来，只有当对应的操作准备好的时候，才开始执行。NIO这种将IO操作拆分成了更小粒度的操作，这便是Reactor模式。

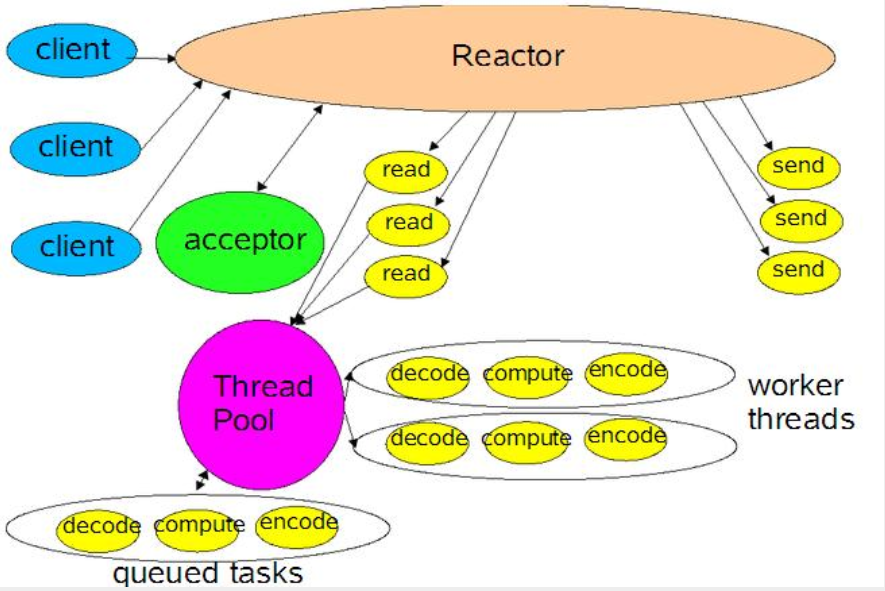
几乎所有的网络连接都会经过 读请求内容—>解码—>计算处理—>编码回复—>回复 的过程。

阻塞IO模型：



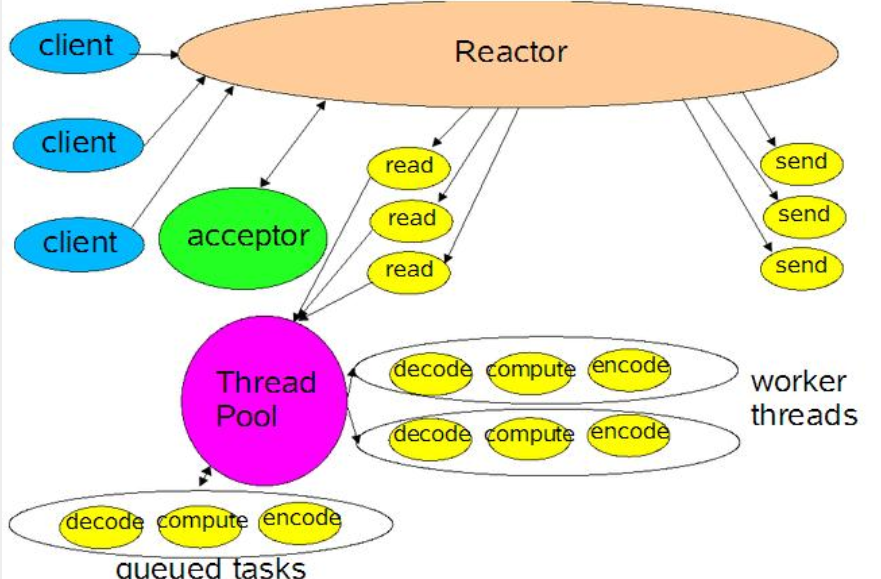
这种模型由于IO在阻塞时会一直等待，因此在用户负载增加时，性能下降的非常快。server导致阻塞的原因：1. serversocket的accept方法，阻塞等待client连接，直到client连接成功。2. 线程从socket inputstream读入数据，会进入阻塞状态，直到全部数据读完。3. 线程向socket outputstream写入数据，会阻塞直到全部数据写完。

非阻塞IO模型：



Reactor：负责响应IO事件，当检测到一个新的事件，将其发送给相应的Handler去处理。Handler：负责处理非阻塞的行为，标识系统管理的资源；同时将handler与事件绑定。acceptor为单个线程，需要处理accept连接，同时发送请求到处理器中。

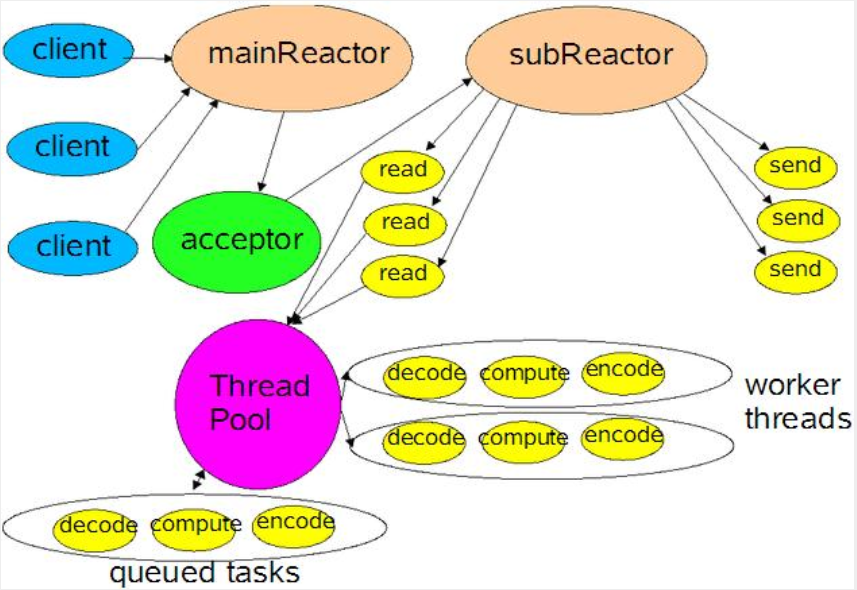
非阻塞IO模型改进版1：



由于handler负责处理具体的IO事件，因此为它提供线程池来提高效率。

Acceptor仍然为单线程，因为它任务很轻，仅仅需要接收连接并给Reactor。

非阻塞IO模型改进版2：



增加reactor来充分利用多核CPU资源，不适用于单核CPU。

Reactor模式的NIO总结：

将接收到的一个新连接交给reactor处理(判断具体是哪种事件)，reactor再从线程池拿出一个线程去处理具体的事件。

# 结合Tomcat看reactor

Tomcat在9.0.x版本使用了reactor模式的NIO。Netty和redis也使用了该模式。

Tomcat下的acceptor对应于reactor的acceptor。

Tomcat下的poller对应于Reactor。

# AIO

AIO即BIO2，但是很多主流框架没有使用AIO，因为在linux下AIO不比BIO快多少，在windows下倒是有比linux更好的性能提升。

AIO是一种基于回调机制的异步非阻塞IO，底层依赖JVM自动构造管理的线程池。它通过新建CompletionHandler子类对象，重写其中的completed(…)和failed(..)方法来实现异步非阻塞读写功能。

AsynchronousServerSocketChannel 或 AsynchronousSocketChannel 的 open() 方法的时候，相应的 channel 就属于默认的 group，这个 group 由 JVM 自动构造并管理。