# 4种I/O的对比

## 概念

### 1.1异步非阻塞I/O

JDK1.7提供NIO2.0新增了异步套接字通道，它才是真正的异步I/O，在异步I/O操作的时候可以传递信号变量，当操作完成之后会回调相关的方法，异步I/O也称为AIO.

NIO类库支持非阻塞读写操作，相比于之前的同步阻塞读写，它是异步的。

### 1.2多路复用器Selector

NIO的实现关键是多路复用I/O技术，多路复用的核心就是通过Selector来轮询注册在其上的Channel,当发现某个或者多个Channel处于就绪状态后，从阻塞状态返回就绪的Channel的选择键集合，进行I/O操作。由于多路复用器是NIO实现非阻塞I/O的关键，它又主要通过Selector实现，所以将Selector称为多路复用器（其他也称为选择器）

### 1.3伪异步I/O

为解决一连接一线程问题。通过线程池做缓冲区的做法。称为伪异步I/O

## 2不同I/O模型对比





## 3选择Netty的理由

Netty对原生NIO API封装，处理了epoll bug (导致Selector空轮询，最终至CPU 100%),性能好，稳定。

原生AIP复杂，需要掌握线程NIO等知识。

同步阻塞I/O，伪异步I/O，非阻塞I/O(NIO)，异步I/O(AIO)

# NIO进行服务端开发的步骤：

1. 创建ServerSocketChannel，配置它为非阻塞模式；
2. 绑定监听，配置TCP参数，例如backlog大小；
3. 创建一个独立的I/O线程，用于轮询多路复用器Selector；
4. 创建Selector，将之前创建的ServerSocketChannel注册到Selector上，监听SelectionKey.ACCEPT;
5. 启动I/O线程，在循环体中执行Selector.select()方法，轮询就绪的Channel；
6. 当轮询到就绪状态的Channel，需要对其进行判断，如果是OP\_ACCEPT状态，说明是新的客户端接入，则盗用ServerSocketChannel.accept()方法接收新的客户端；
7. 设置新接入的客户端链路SocketChannel为非阻塞模式，配置其他TCP参数；
8. 将SocketChannel注册到Selector，监听OP\_READ操作位；
9. 如果轮询到Channel为OP\_READ,则说明SocketChannel中有新就绪的数据包需要读取，则构造ByteBuffer对象，读取数据包。
10. 如果轮询到Channel为OP\_WRITE,则说明数据没有发送完成，需要继续发送；