同步与异步（synchronous/asynchronous）：同步是一种可靠的有序运行机制，当我们进行同步操作时，后续的任务是等待当前调用返回，才会进行下一步；而异步则相反，其他任务不需要等待当前调用返回，通常依靠事件、回调等机制来实现任务间次序关系

阻塞与非阻塞：在进行阻塞操作时，当前线程会处于阻塞状态，无法从事其他任务，只有当条件就绪才能继续，比如ServerSocket新连接建立完毕，或者数据读取、写入操作完成；而非阻塞则是不管IO操作是否结束，直接返回，相应操作在后台继续处理

同步和异步的概念：实际的I/O操作

同步是用户线程发起I/O请求后需要等待或者轮询内核I/O操作完成后才能继续执行

异步是用户线程发起I/O请求后仍需要继续执行，当内核I/O操作完成后会通知用户线程，或者调用用户线程注册的回调函数

阻塞和非阻塞的概念：发起I/O请求

阻塞是指I/O操作需要彻底完成后才能返回用户空间

非阻塞是指I/O操作被调用后立即返回一个状态值，无需等I/O操作彻底完成

BIO：同步阻塞式IO，服务器实现模式为一个连接一个线程，即客户端有连接请求时服务器端就需要启动一个线程进行处理，如果这个连接不做任何事情会造成不必要的线程开销，当然可以通过线程池机制改善。

NIO：同步非阻塞式IO，服务器实现模式为一个请求一个线程，即客户端发送的连接请求都会注册到多路复用器上，多路复用器轮询到连接有I/O请求时才启动一个线程进行处理。

AIO(NIO.2)：异步非阻塞式IO，服务器实现模式为一个有效请求一个线程，客户端的I/O请求都是由OS先完成了再通知服务器应用去启动线程进行处理。

NIO: 三个主要组成部分：Channel（通道）、Buffer（缓冲区）、Selector（选择器）

各自应用场景

到这里你也许已经发现，一旦有请求到来(不管是几个同时到还是只有一个到)，都会调用对应IO处理函数处理，所以：

（1）NIO适合处理连接数目特别多，但是连接比较短（轻操作）的场景，Jetty，Mina，ZooKeeper等都是基于java nio实现。

（2）BIO方式适用于连接数目比较小且固定的场景，这种方式对服务器资源要求比较高，并发局限于应用中。

（3）AIO新的IO2.0，即NIO2.0，jdk1.7开始应用，叫做异步不阻塞的IO。AIO引入异常通道的概念，采用了Proactor模式，简化了程序编写，一个有效的请求才启动一个线程，它的特点是先由操作系统完成后才通知服务端程序启动线程去处理，一般适用于连接数较多且连接时间长的应用。