IO

如何区分“同步/异步”和“阻塞/非阻塞”：

同步/异步是从行为角度描述事物的，而阻塞和非阻塞描述的是当前事物的状态（等待调用结果时的状态）

BIO（Blocking IO）

同步阻塞模式，数据的读取写入必须阻塞在一个线程内等待其完成。

- 传统BIO

  一请求一应答模型，一旦接收到一个连接请求，就可以建立通信套接字进行读写操作，此时不能在接受其他客户端连接请求，只能等待同当前连接的客户端的操作执行完成，不过可以通过多线程同时处理多个客户端的请求。

  如果客户端并发访问量增加，这种模型线程创建和销毁、切换成本很高，可能导致堆栈溢出，创建新线程失败等问题。

- 伪异步IO

  为了解决阻塞IO面临的一个链路需要一个线程处理的问题，进行了优化——后端通过一个线程池来处理多个客户端的请求接入。JDK的线程池维护一个消息队列和N个活跃线程，对消息队列中的任务进行处理，由于线程池可以设置消息队列的大小和线程数，资源占用是可控的。

NIO（New IO）

同步非阻塞的I/O模型。N代表Non-blocking。

NIO特性/NIO与IO区别：

NIO流是非阻塞IO而IO流是阻塞IO。

1. Non-blocking IO（非阻塞IO）

   NIO使我们可以进行非阻塞IO操作。比如单线程从通道读取数据到buffer，可以同时继续做别的事情，当数据读取到buffer中后，线程再继续处理数据。

   IO的各种流是阻塞的，当一个线程调用了read()或write()时，该线程被阻塞，直到数据被读取或完全写入。

2. Buffer（缓冲区）

   IO面向流（Stream Oriented），而NIO面向缓冲区（Buffer oriented）

   NIO库中，所有数据都是用缓冲区处理的。读取数据时，直接读到缓冲区；写入数据时，写入到缓冲区中。

3. Channel（通道）

   NIO通过Channel进行读写。

   通道是双向的，可读也可写，而流的读写是单向的。无论读写，通道只能和Buffer交互，因为Buffer，通道可以异步地读写。

4. Selector（选择器）

   NIO有选择器，而IO没有。

   选择器用于单个线程处理多个通道。因此它需要较少的线程来处理这些通道。线程之间的切换对于操作系统来说是昂贵的。 因此，为了提高系统效率选择器是有用的。

NIO读数据和写数据方式

所有IO都是从Channel开始：

- 从通道进行数据读取：创建一个缓冲区，然后请求通道读取数据
- 从通道进行数据写入：创建一个缓冲区，填充数据，并要求通道写入数据

NIO核心组件：

- Channel 通道
- Buffer 缓冲
- Selector 选择器

AIO（Asynchronous IO）

也就是NIO2，异步非阻塞的IO模型。异步IO是基于**事件和回调机制**实现的。应用操作之后不会阻塞在那里，当后台处理完成，操作系统会通知相应的线程进行后续操作。

同步IO模型：

阻塞IO模型、非阻塞IO模型、IO复用模型、信号驱动IO模型。因为真正数据拷贝过程都是同步进行的。类比钓鱼，1.鱼咬钩（数据准备）2.把鱼钓起来放到竹篓里（数据拷贝）。无论以上哪种钓鱼方式，第二步都是需要人主动去做的，并不是鱼竿自己完成的，所以这个钓鱼过程其实还是同步完成的。

异步IO模型：

应用进程把IO请求给内核后，完全由内核去操作文件拷贝。内核完成相关操作后，会发信号告诉应用进程本次IO已经完成。