# 【第02章-NIO入门】【04-四种IO对比】

## 异步非阻塞IO

很多人将JDK 1.4 提供的NIO 框架称为异步非阻塞IO， 但是，如果严格按照UNIX网络编程模型和lJDK 的实现进行区分， 实际上它只能被称为非阻塞IO ， 不能叫异步非阻塞 。在早期的JDK 1.4 和1.5 update 10版本之前， JDK 的Selector 基于select/poll模型实现，它是基于IO复用技术的非阻塞IO，不是异步IO。在JDK 1.5 update I 0 和Linux core2.6 以上版本， Sun 优化了Selector 的实现，它在底层使用epoll 替换了select/poll ， 上层的API 并没有变化，可以认为是JDK NIO 的一次性能优化，但是它仍旧没有改变IO模型。由JDKl. 7 提供的NIO 2.0 新增了异步的套接字通道，它是真正的异步IO， 在异步IO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IO也被称为AIO 。

## 多路复用器Selector

通过Java NIO 的实现关键是多路复用IO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IO操作。由于多路复用器是NIO实现非阻塞IO的关键，它又是主要通过Selector 实现的。

## 伪异步IO

伪异步IO的概念完全来源于实践。在JDK NIO 编程没有流行之前，为了解决Tomcat通信线程同步IO导致业务线程被挂住的问题，大家想到了一个办法： 在通信线程和业务线程之间做个缓冲区，这个缓冲区用于隔离IO线程和业务线程间的直接访问，这样业务线程就不会被IO线程阻塞。而对于后端的业务侧来说，将消息或者Task放到线程池后就返回了，它不再直接访问IO 线程或者进行IO读写，这样也就不会被同步阻塞。类似的设计还包括前端启动一组线程，将接收的客户端封装成Task，放到后端的线程池执行，用于解决一连接一线程问题。

## 不同IO模型对比

几种IO模型的功能和特性对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 同步阻塞IO( BIO) | 伪异步IO | 非阻塞IO ( NIO) | 异步IO( AIO ) |
| 客户端个数：IO 线程 | 1:1 | M:N | M:1 | M:0（不需要启动IO线程，被动回调） |
| IO类型（阻塞） | 阻塞IO | 阻塞IO | 非阻塞IO | 非阻塞IO |
| IO类型（同步） | 同步IO | 同步IO | 同步IO（多路复用） | 异步IO |
| API使用难度 | 简单 | 简单 | 非常复杂 | 复杂 |
| 调试难度 | 简单 | 简单 | 复杂 | 复杂 |
| 可靠性 | 非常差 | 差 | 高 | 高 |
| 吞吐量 | 低 | 中 | 高 | 高 |