

（**同步和异步是针对应用程序和内核的交互而言的，同步指的是用户进程触发IO 操作并等待或者轮询的去查看IO 操作是否就绪，而异步是指用户进程触发IO 操作以后便开始做自己的事情，而当IO 操作已经完成的时候会得到IO 完成的通知。**）

（**阻塞和非阻塞是针对于进程在访问数据的时候，根据IO操作的就绪状态来采取的不同方式，说白了是一种读取或者写入操作方法的实现方式，阻塞方式下读取或者写入函数将一直等待，而非阻塞方式下，读取或者写入方法会立即返回一个状态值**。）

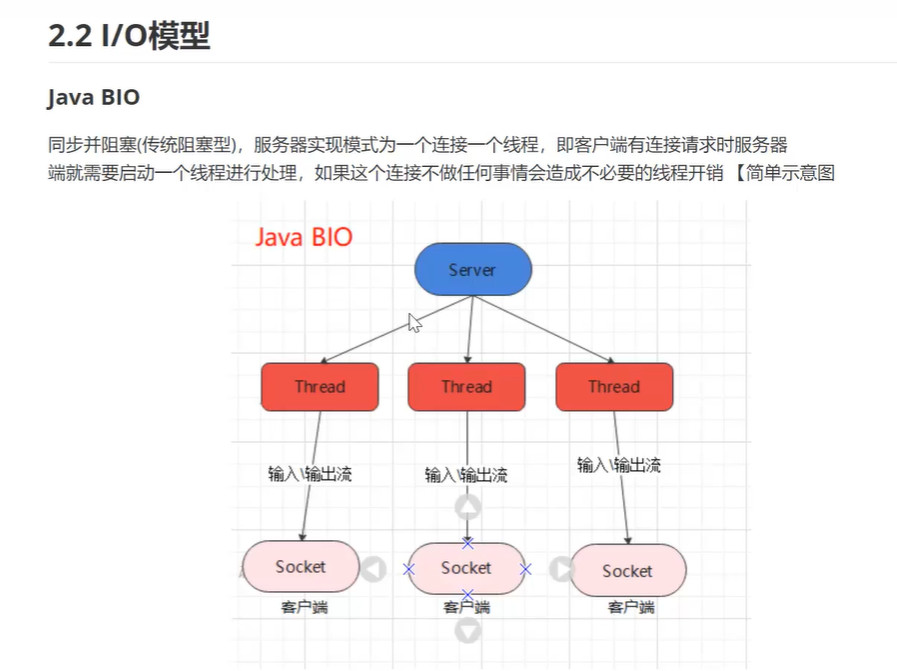
（同步和异步是指多个请求顺序执行还是并行，阻塞和非阻塞是指在执行过程中是执行到底还是会被滞留在某处）

（为什么会觉得同步和阻塞是一个概念？是因为两者引起的现象会相同，等待。比如只有一个线程，多个请求那么就只能同步，对于没有资源的请求就是等待现象，也被人成为阻塞现象；而阻塞其实是一个广义的概念，往往是指某种条件不满足而导致不能继续，往往表现为等待，所以其实上说阻塞一种结果，同步是一种原因，所以有同步阻塞（因为请求多个）的说法，这个说法和JAVA中IO不一样，我认为JavaIO中的同步阻塞应该叫做同步阻塞读阻塞。）（那么反过来异步与非阻塞就好理解了，异步，资源充足，同时处理；非阻塞，遇见非必要结果，可以直接返回，继续执行，不用等待该条件。）

（所以同步异步还有个特点，面对临界资源且必须。阻塞非阻塞也是因为临界资源问题，但是还包含非必须部分。）

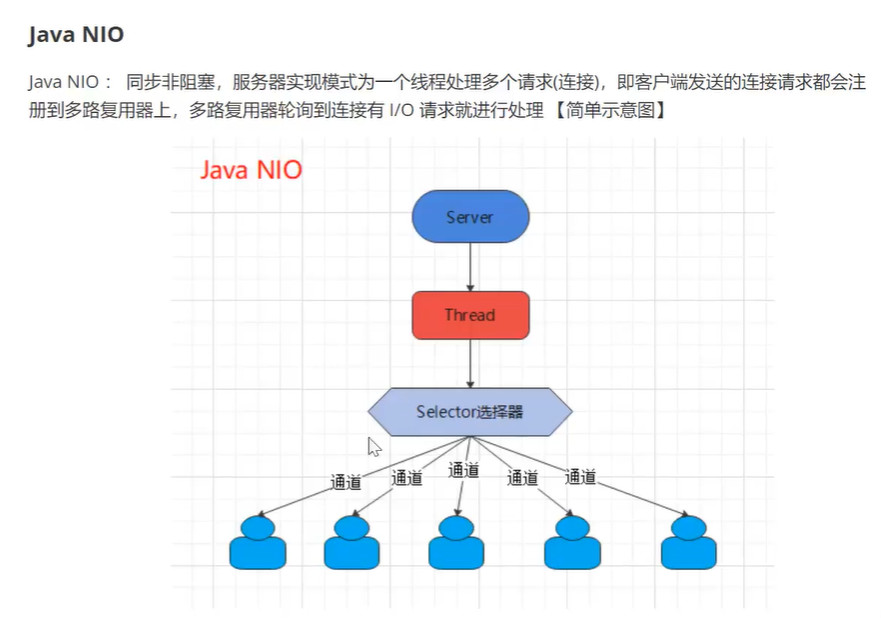
<https://zhuanlan.zhihu.com/p/66148226>等待

<https://www.cnblogs.com/Rivend/p/12065474.html>



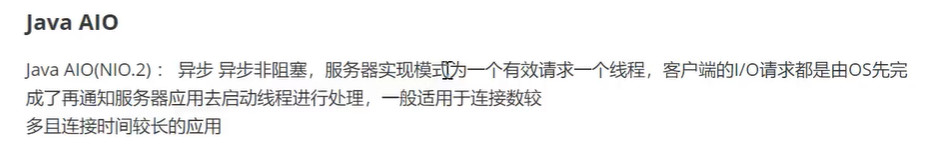
**（该模型在读取数据时，客户端与服务器端是同步的，所以只要有一端要进行数据的读取，另一方就必须进行等待）**

**(同步阻塞：当有多个请求时，如果没有多线程，就只能一个请求处理完，再下一个请求，而且在处理请求过程中时执行IO操作时数据就阻塞)**



（该模型是一个线程处理多个**连接**请求，注意网络中一般是突发式数据交流。）

（同步非阻塞：如果没有轮训器，仍然是请求来时需要挨个进行处理，但是现在有了轮训器我们是轮询监听，其实也是一种顺序遍历过程，如果没有遍历到你，你就得等待。但是只要轮到你，就不会滞留一路执行到底。所以他不能同时处理多个请求，但是处理请求过程的IO操作能直接返回不用阻塞。所以）



（异步非阻塞，多个请求能够同时被处理而且处理过程中对IO资源可以直接返回不用阻塞）

