链滴

BIO NIO AIO

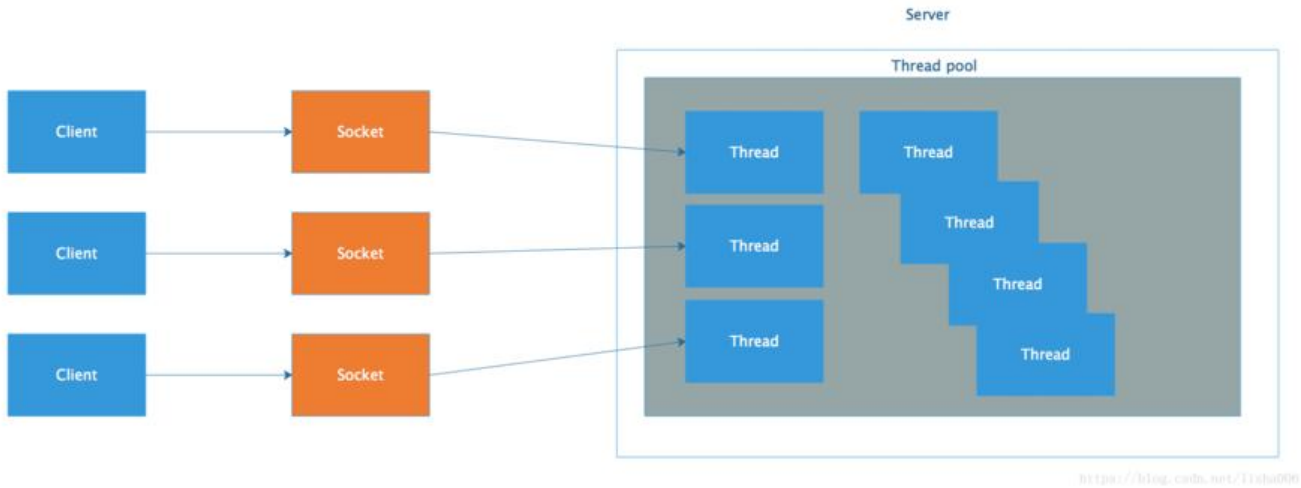
作者：gaga

原文链接：https://ld246.com/article/1583247270476

来源网站：链滴

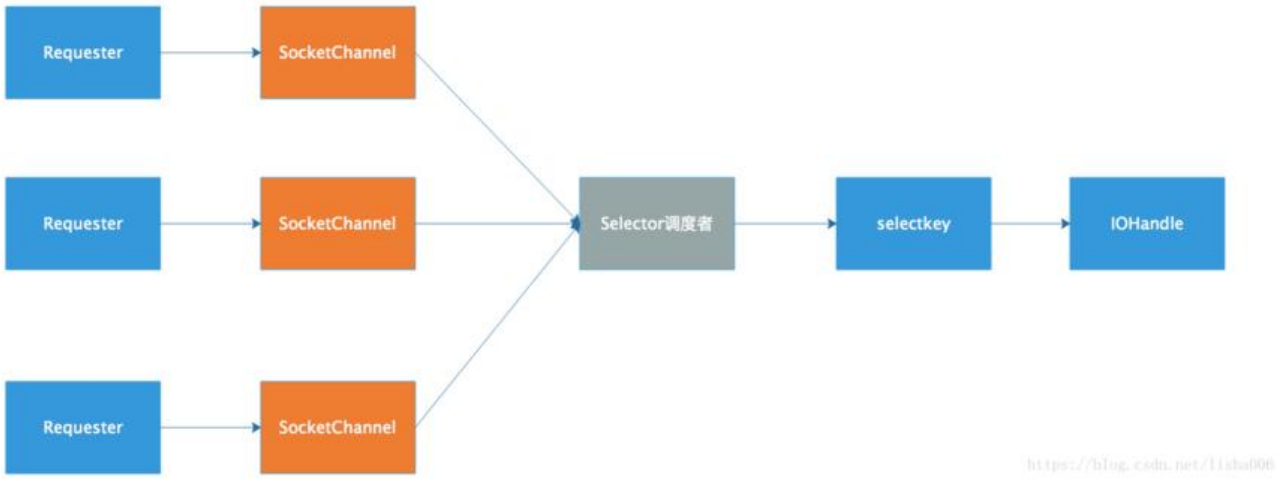
一、同步阻塞 IO（Java BIO）：

同步并阻塞，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程进行处理。客户端发送数据后响应返回，如果客户端不发信息，线程阻塞挂起。



二、同步非阻塞 IO(Java NIO)：

对比 BIO 的同步阻塞 IO 操作，实际上 NIO 是同步非阻塞 IO，一个线程在同步的进行轮询检查，Selctor 不断轮询注册在其上的 Channel，某个 Channel 上面发生读写连接请求，这个 Channel 就处于绪状态，被 Selector 轮询出来，然后通过 SelectionKey 可以获取就绪 Channel 的集合，进行后续的/O 操作。这样提高了线程利用率，同时也能支持更多客户端。



三、 （Java AIO(NIO.2)）异步非阻塞 IO:

AIO 不需要通过多路复用器对注册的通道进行轮询操作即可实现异步读写。什么意思呢？NIO 采用轮的方式，一直在轮询的询问 stream 中数据是否准备就绪，如果准备就绪发起处理。但是 AIO 就不需了，AIO 框架在 windows 下使用 windows IOCP 技术，在 Linux 下使用 epoll 多路复用 IO 技术模异步 IO， 即：应用程序向操作系统注册 IO 监听，然后继续做自己的事情。操作系统发生 IO 事件，且准备好数据后，在主动通知应用程序，触发相应的函数（这就是一种以订阅者模式进行的改造）。于应用程序不是“轮询”方式而是订阅-通知方式，所以不再需要 selector 轮询，由 channel 通道直到操作系统注册监听。

<pre class="vditor-reset" placeholder="" contenteditable="true" spellcheck="false"><p data

原文链接：BIO NIO AIO

block="0">

</pre>

四、NIO 和 AIO 对比

NIO：会等数据准备好后，再交由应用进行处理，数据的读取/写入过程依然在应用线程中完成，只是等待的时间剥离到单独的线程中去，节省了数据准备时间，因为多路复用机制，Selector 会得到复用对于那些读写过程时间长的，NIO 就不太适合。

AIO：读完（内核内存拷贝到用户内存）了系统再通知应用，使用回调函数，进行业务处理，AIO 能胜任那些重量级，读写过程长的任务。