1.同步：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。排队

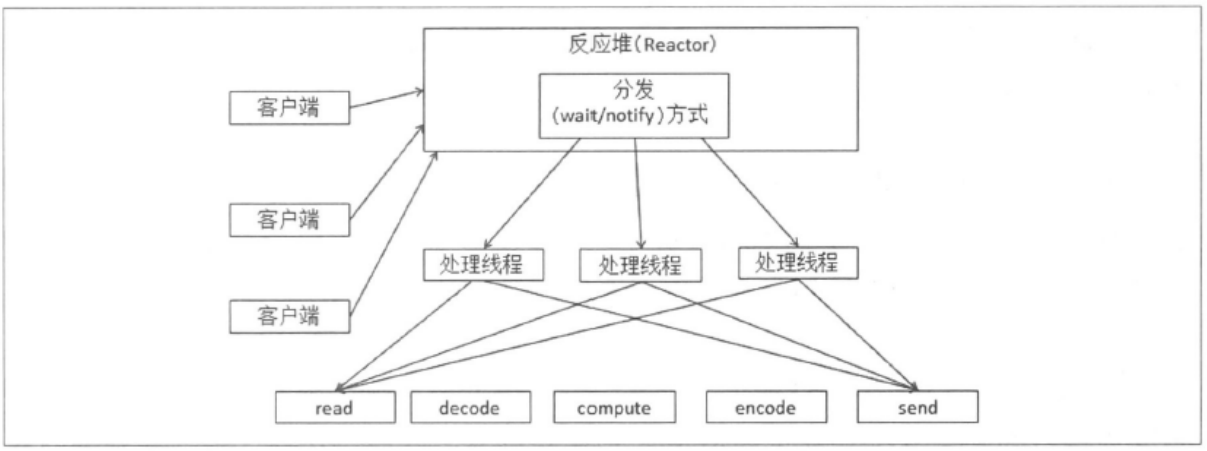
2.异步：用户触发IO操作以后,可以干别的事，IO操作完成以后再通知当前线程。

3.阻塞：当试图进读写文件的时候，发现不可读取或没东西读，则进入等待状态直到可读。

4.非阻塞：用户进程访问数据时，会马上返回一个状态值(可读不可读)，（使用非阻塞IO时，如果不能读写Java调用会马上返回，当IO事件分发器会通知可读写时再继续进行读写，不断循环直到读写完成）。

BIO(同步阻塞)，每一个socket套接字需要使用一个线程来处理。建立连接、进行读写操作的时候都可能阻塞。在服务器端如果要支持并发的连接时，需要更多的线程。连接不做任何事情的时候会造成不必要的线程开销，可通过线程池来改善。

NIO(同步非阻塞) 基于**事件驱动**，采用的Reactor模式，(Reactor模式首先是**事件驱动的，有一个或多个并发输入源，有一个Service Handler，有多个Request Handlers**；这个Service Handler会同步的将输入的请求（Event）多路复用的分发给相应的Request Handler。)Reactor会处理所有客户端的Socket套接字的事件，然后派发到不同的线程中。这样就解决了BIO中为了支撑更多的Socket套接字而需要更多的线程。



AIO（异步非阻塞）AIO采用了Proactor模式，AIO与NIO的不同之处在于当AIO在进行读写操作时，不用先等通知，可直接调用相应的read/write方法，这两种方法均为异步的。对于读操作而言，当有流可读取时，操作系统会将可读的流传入read方法的缓冲区，并通知应用程序；对于写操作而言，当操作系统将write方法传递的流写入完毕时，操作系统主动通知应用程序,而NIO的通知是发生在动作之前的，是在可读、写的时候，Selector发现了这些事件后调用Handler处理