http://daimojingdeyu.iteye.com/blog/828696

Reactor这个词译成汉语还真没有什么合适的，很多地方叫反应器模式，但更多好像就直接叫reactor模式了，其实我觉着叫应答者模式更好理解一些。通过了解，这个模式更像一个侍卫，一直在等待你的召唤，或者叫召唤兽。

并发系统常使用reactor模式，代替常用的多线程的处理方式，节省系统的资源，提高系统的吞吐量。

先用比较直观的方式来介绍一下这种方式的优点，通过和常用的多线程方式比较一下，可能更好理解。
以一个餐饮为例，每一个人来就餐就是一个事件，他会先看一下菜单，然后点餐。就像一个网站会有很多的请求，要求服务器做一些事情。处理这些就餐事件的就需要我们的服务人员了。

在多线程处理的方式会是这样的:
一个人来就餐，一个服务员去服务，然后客人会看菜单，点菜。 服务员将菜单给后厨。
二个人来就餐，二个服务员去服务……
五个人来就餐，五个服务员去服务……

这个就是多线程的处理方式，一个事件到来，就会有一个线程服务。很显然这种方式在人少的情况下会有很好的用户体验，每个客人都感觉自己是VIP，专人服务的。如果餐厅一直这样同一时间最多来5个客人，这家餐厅是可以很好的服务下去的。

来了一个好消息，因为这家店的服务好，吃饭的人多了起来。同一时间会来10个客人，老板很开心，但是只有5个服务员，这样就不能一对一服务了，有些客人就要没有人管了。老板就又请了5个服务员，现在好了，又能每个人都受VIP待遇了。

越来越多的人对这家餐厅满意，客源又多了，同时来吃饭的人到了20人，老板高兴不起来了，再请服务员吧，占地方不说，还要开工钱，再请人就攒不到钱了。怎么办呢？老板想了想，10个服务员对付20个客人也是能对付过来的，服务员勤快点就好了，伺候完一个客人马上伺候另外一个，还是来得及的。综合考虑了一下，老板决定就使用10个服务人员的线程池啦~~~

但是这样有一个比较严重的缺点就是，如果正在接受服务员服务的客人点菜很慢，其

他的客人可能就要等好长时间了。有些火爆脾气的客人可能就等不了走人了。

Reactor如何处理这个问题呢：
老板后来发现，客人点菜比较慢，大部服务员都在等着客人点菜，其实干的活不是太多。老板能当老板当然有点不一样的地方，终于发现了一个新的方法，那就是：当客人点菜的时候，服务员就可以去招呼其他客人了，等客人点好了菜，直接招呼一声“服务员”，马上就有个服务员过去服务。嘿嘿，然后在老板有了这个新的方法之后，就进行了一次裁员，只留了一个服务员！这就是用单个线程来做多线程的事。

实际的餐馆都是用的Reactor模式在服务。一些设计的模型其实都是从生活中来的。

Reactor模式主要是提高系统的吞吐量，在有限的资源下处理更多的事情。

在单核的机上，多线程并不能提高系统的性能，除非在有一些阻塞的情况发生。否则线程切换的开销会使处理的速度变慢。就像你一个人做两件事情，1、削一个苹果。2、切一个西瓜。那你可以一件一件的做，我想你也会一件一件的做。如果这个时候你使用多线程，一会儿削苹果，一会切西瓜，可以相像究竟是哪个速度快。这也就是说为什么在单核机上多线程来处理可能会更慢。

但当有阻碍操作发生时，多线程的优势才会显示出来，现在你有另外两件事情去做，1、削一个苹果。2、烧一壶开水。我想没有人会去做完一件再做另一件，你肯定会一边烧水，一边就把苹果削了。

理论的东西就不多讲了，请大家参考一下附件《reactor-siemens.pdf》。图比较多，E文不好也可以看懂的。