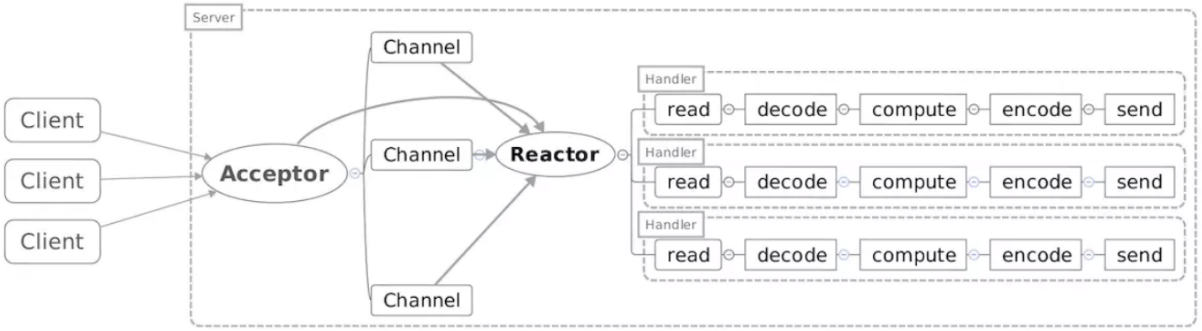
Reactor 的特点是 I/O 多路复用和事件驱动

组件

- Reactor:Reactor是IO事件的派发者。
- Acceptor:Acceptor接受client连接，建立对应client的Handler，并向Reactor注册此Handler。
- Handler:和一个client通讯的实体，按这样的过程实现业务的处理。一般在基本的Handler基础上还会有更进一步的层次划分， 用来抽象诸如decode，process和encoder这些过程。比如对Web Server而言，decode通常是HTTP请求的解析， process的过程会进一步涉及到Listener和Servlet的调用。业务逻辑的处理在Reactor模式里被分散的IO事件所打破， 所以Handler需要有适当的机制在所需的信息还不全（读到一半）的时候保存上下文，并在下一次IO事件到来的时候（另一半可读了）能继续中断的处理。为了简化设计，Handler通常被设计成状态机，按GoF的state pattern来实现。

- Reactor：相当于有分发功能的Selector
- Acceptor：NIO中建立连接的那个判断分支
- Handler：消息读写处理等操作类

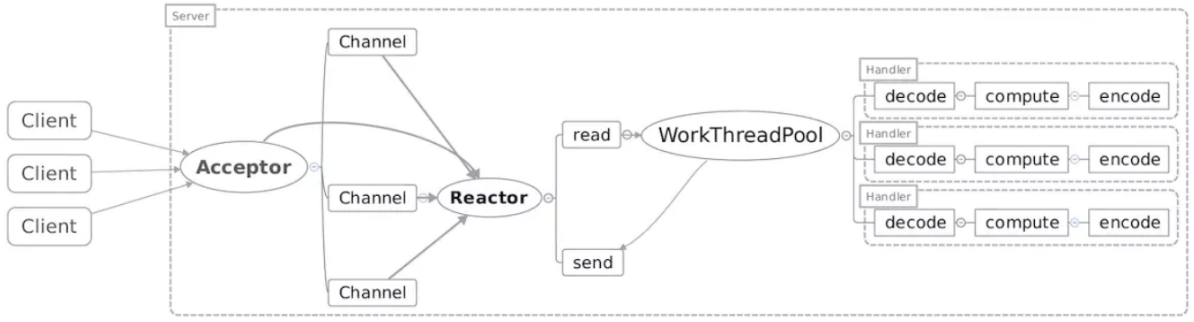
Reactor单线程模型



这个模型和上面的NIO流程很类似，只是将消息相关处理独立到了Handler中去了！

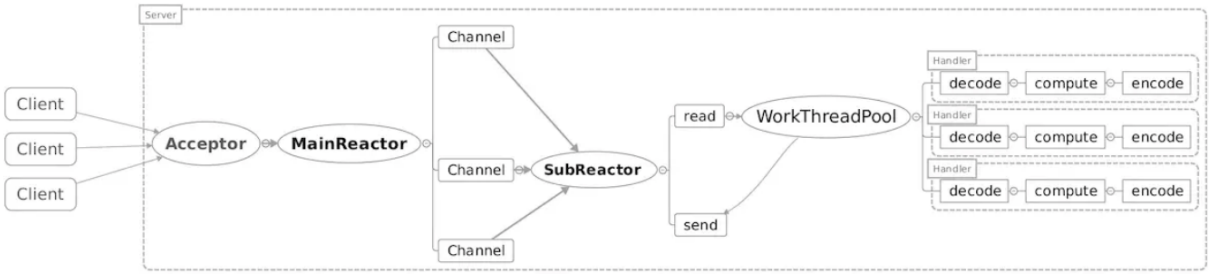
虽然上面说到NIO一个线程就可以支持所有的IO处理。但是瓶颈也是显而易见的！我们看一个客户端的情况，如果这个客户端多次进行请求，如果在Handler中的处理速度较慢，那么后续的客户端请求都会被积压，导致响应变慢！所以引入了Reactor多线程模型!

Reactor多线程模型



Reactor多线程模型就是将Handler中的IO操作和非IO操作分开，操作IO的线程称为IO线程，非IO操作的线程称为工作线程!这样的话，客户端的请求会直接被丢到线程池中，客户端发送请求就不会堵塞！
但是当用户进一步增加的时候，Reactor会出现瓶颈！因为Reactor既要处理IO操作请求，又要响应连接请求！为了分担Reactor的负担，所以引入了主从Reactor模型!

主从Reactor模型



主Reactor用于响应连接请求，从Reactor用于处理IO操作请求！

参考文章：
https://www.jianshu.com/p/2461535c38f3