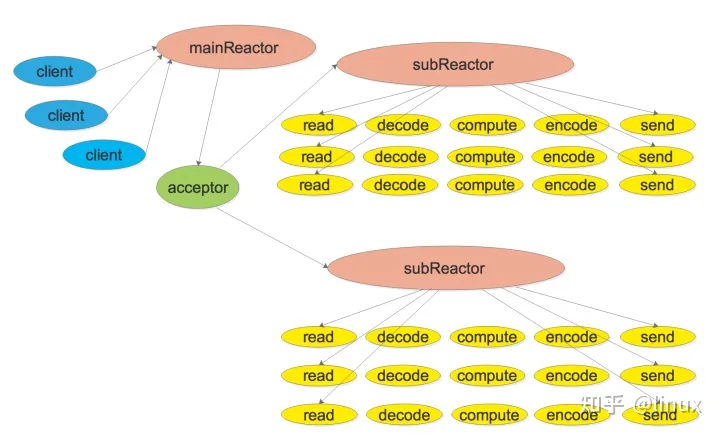
孙恒康第二简历：

### 异常汇聚系统：

#### 你说借鉴了muduo库的设计，那你介绍一下muduo库吧

Muduo库有三个核心组件支撑一个reactor实现 [持续] 的 [监听] 一组fd，并根据每个fd上发生的事件 [调用] 相应的处理函数。这三个组件分别是Channel类、Poller/EpollPoller类以及EventLoop类。



1. Channel

Channel类则封装了一个 [fd] 和这个 [fd感兴趣事件] 以及事件监听器监听到 [该fd实际发生的事件]，以及发生的个时间我要怎么处理这个数据的[Function]同时Channel类还提供了设置该fd的感兴趣事件，以及将该fd及其感兴趣事件注册到事件监听器或从事件监听器上移除，以及保存了该fd的每种事件对应的处理函数。

下面是他的成员变量：

* fd\_这个Channel对象照看的文件描述符
* int events\_代表fd感兴趣的事件类型集合 int
* revents\_代表事件监听器实际监听到该fd发生的事件类型集合，当事件监听器监听到一个fd发生了什么事件。
* EventLoop\* loop：这个fd属于哪个EventLoop对象，这个暂时不解释。
* read\_callback\_、write\_callback\_、close\_callback\_、error\_callback\_：这些是std::function类型，代表着这个Channel为这个文件描述符保存的各事件类型发生时的处理函数。

1. EpollPoller

负责监听文件描述符事件是否触发以及返回发生事件的文件描述符以及具体事件的模块就是Poller。所以一个Poller对象对应一个事件监听器（这里我不确定要不要把Poller就当作事件监听器）。在multi-reactor模型中，有多少reactor就有多少Poller。

* epollfd\_: 就是用epoll\_create方法返回的epoll句柄，这个是常识。
* channels\_：这个变量是std::unordered\_map<int, Channel\*>类型，负责记录 文件描述符 —> Channel的映射，也帮忙保管所有- 注册在你这个Poller上的Channel。就是一个文件描述符对应的channel
* ownerLoop\_：所属的EventLoop对象，看到后面你懂了。

1. EventLoop

EventLoop就是负责实现“循环”，负责驱动“循环”的重要模块！！Channel和Poller其实相当于EventLoop的手下，EventLoop整合封装了二者并向上提供了更方便的接口来使用。

#### Reactor架构

事件驱动架构（event-driven architecture） 这种架构把要使用 CPU 的内容定义为一个事件 比如网络编程中的客户端接入、数据读取、数据发送、连接关闭 当有客户端接入的事件发生的时候 我们就安排线程来处理接入 处理完了，就将该事件移出线程，等待其他事件的发生