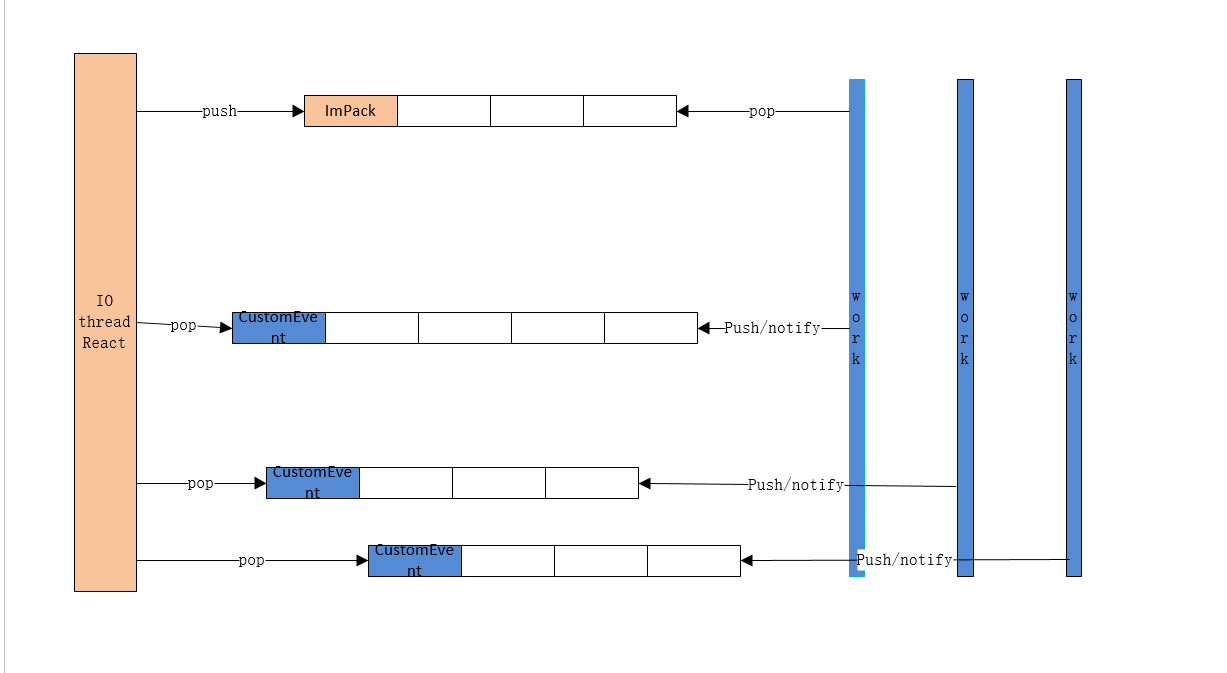
uvclient的程序结构

多线程结构图



基于libuv网络库的c++压测客户端

Io thread: socket线程，负责socket的读写

Woker thread: 工作线程，负责io thread push的业务Impack; work线程处理结果如果还需要io thread做些事情，可以向io 线程投递一个CustomEvent，并通知io thread， io thread根据event区分接下来要处理逻辑

Io thread和woker线程的通讯采用无锁缓冲，缓冲的特点：只允许“单生产者“，可以多消费者。

程序代码结构、及实现细节可以参考readme.txt 和 阅读源码

开展压测业务时，主要是依靠定时器。结合几个容器里的数据：list, map, vector 主要存着当前在线的用户，及建立的连接，和它们的绑定关系。

V1.0

修改代码

1、过程量uv\_connect\_t不保存，建立映射关系时不再是map<uv\_tcp\_t\*, uv\_connect\_t\*>

2、网络层和业务层的关联通过一个map就行map<uv\_tcp\_t\*, UserInfo\*>

3、业务层也可以内置指针关联网络层信息，但业务层何时使用网络层信息并没限定，所以网络层的指针不会也不应该在业务层处理

4、网络层结构体变量uv\_tcp\_t的data字段按理仅仅挂接一个接受缓冲比较合理，但是如果data字段被占用了，那么上边提到的【网络层和业务层的关联】map就得多处声明

5、data字段挂一个全局的【网络层和业务层关联map】也行，避免代码的入侵（不用多处声明这个map）,但缓冲区怎么关联连接就得另想设计了。比如map<uv\_tcp\_t\*, struct UserInfo\_Cache{buf, userInfo}>