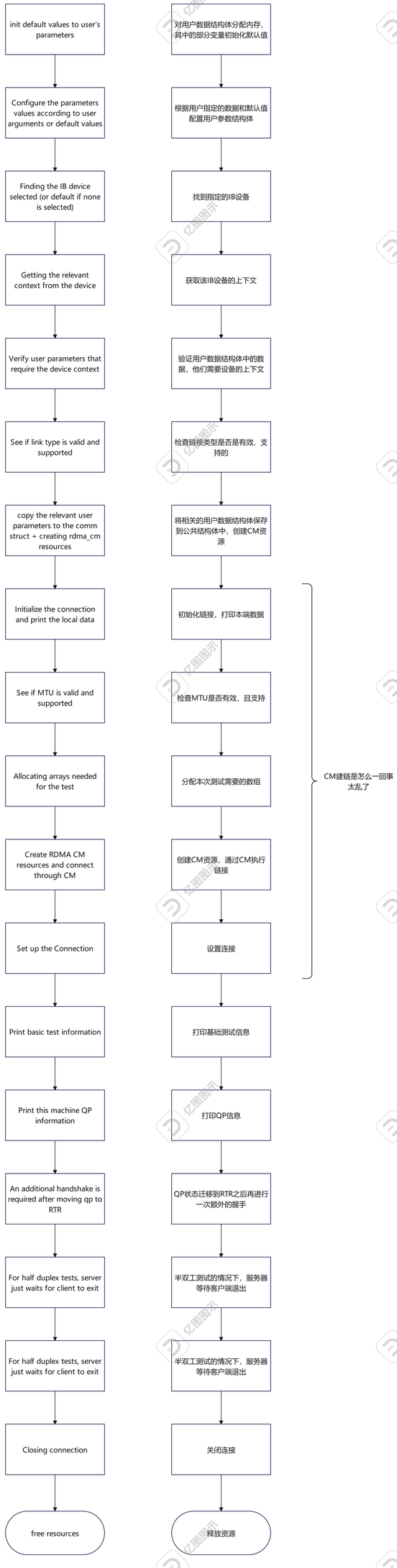
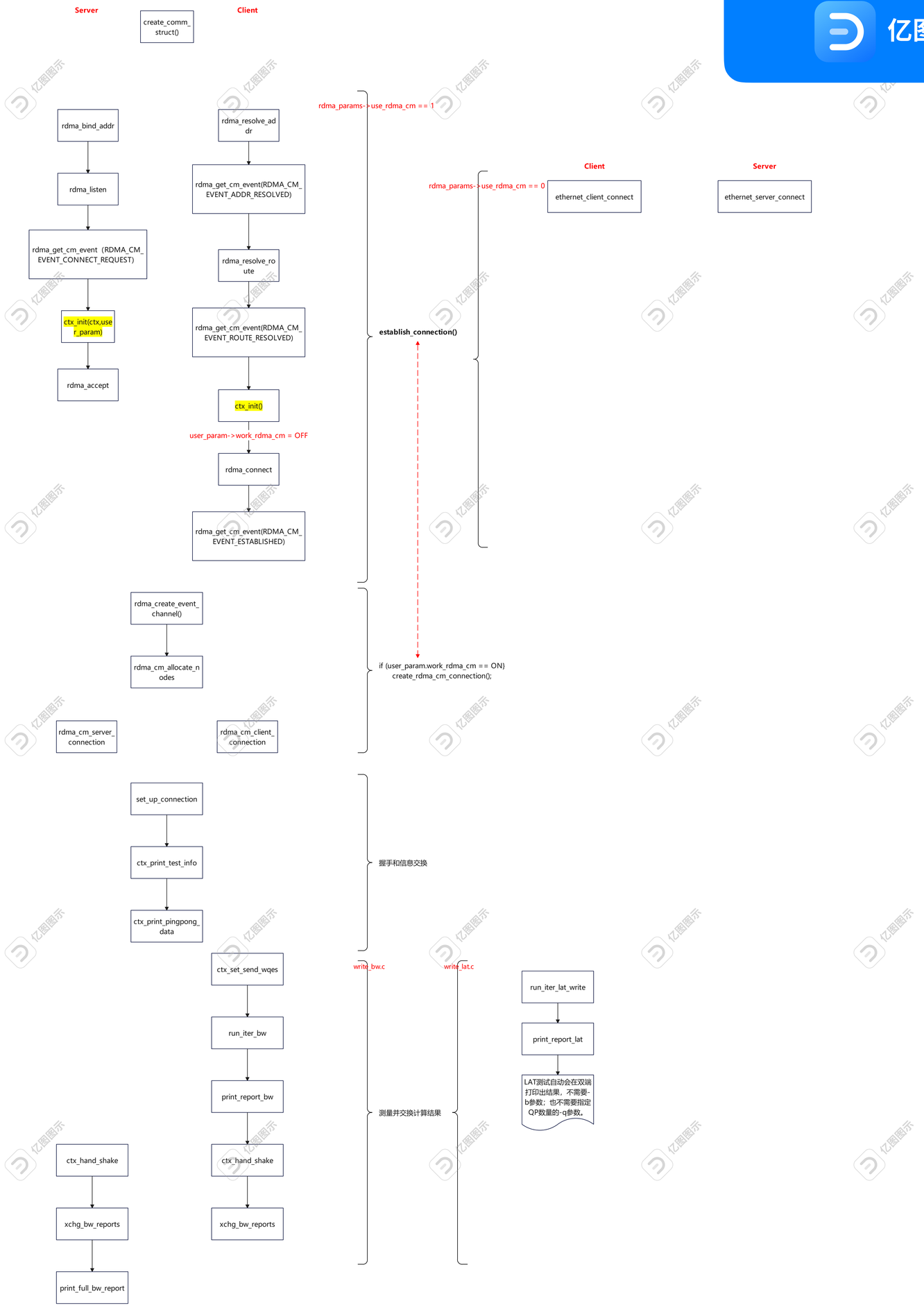
流程比较复杂，丰富！

write_bw.c





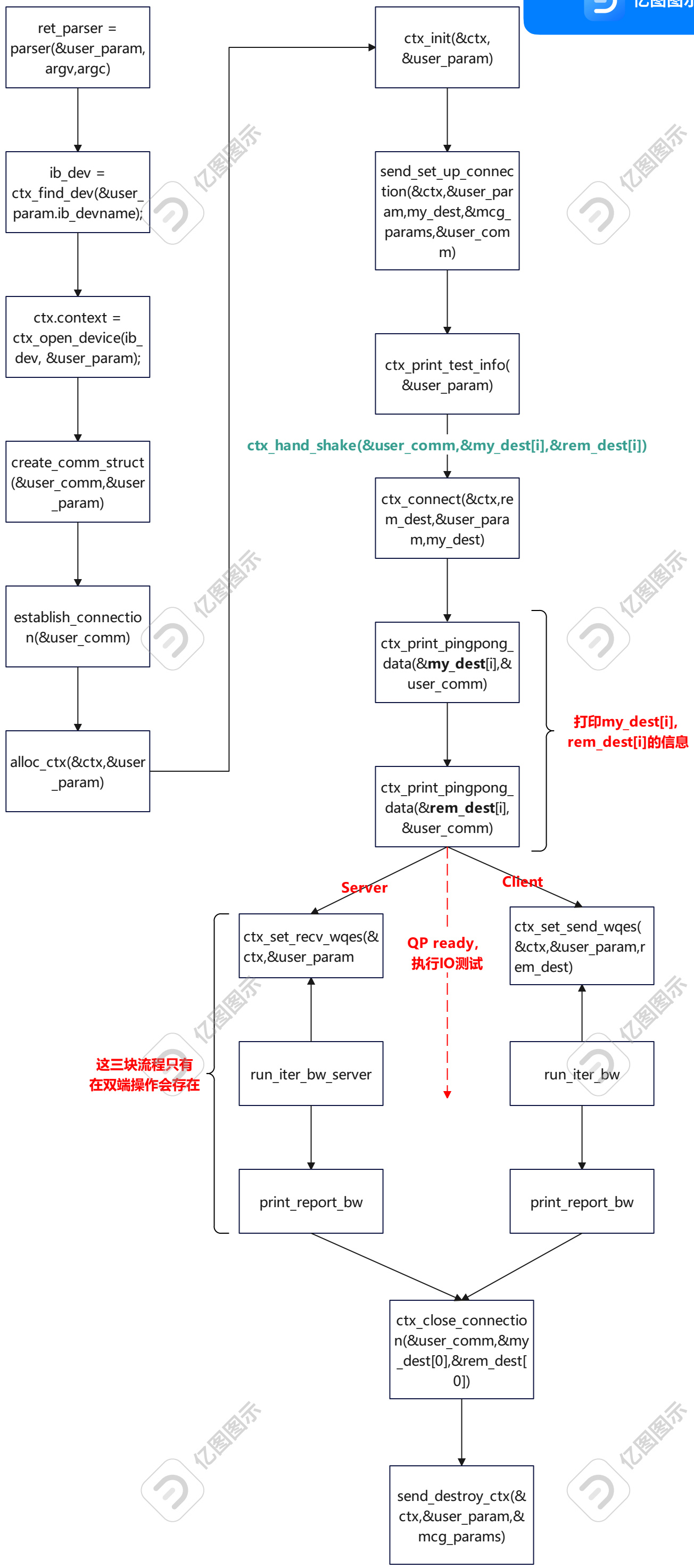
总结：

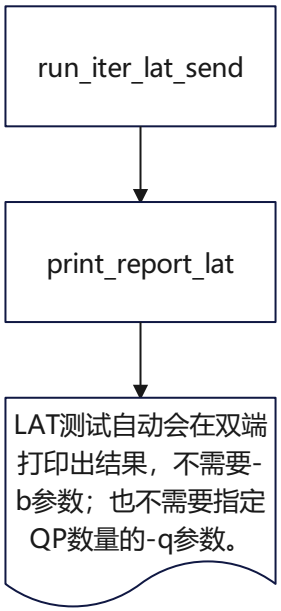
从这write的bw和latency测试我们可以体会到，一个最关键的点是在执行IO和Poll cq的过程中记录下时钟计数，然后才能计算出延迟和带宽。

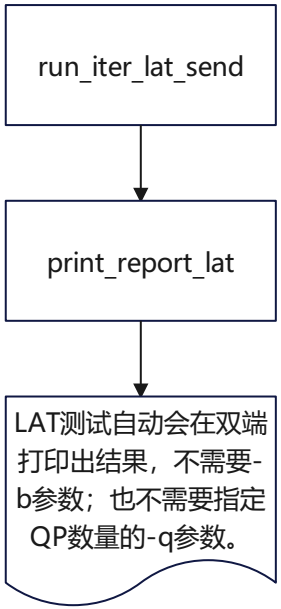
根据计算方式，这里的带宽描述的是对payload的传输，跟网卡的带宽不是一回事，这个工具测试出来的带宽描述的是传输data的能力；时延也是IO和Poll cq的完整流程都结束的平均时间！

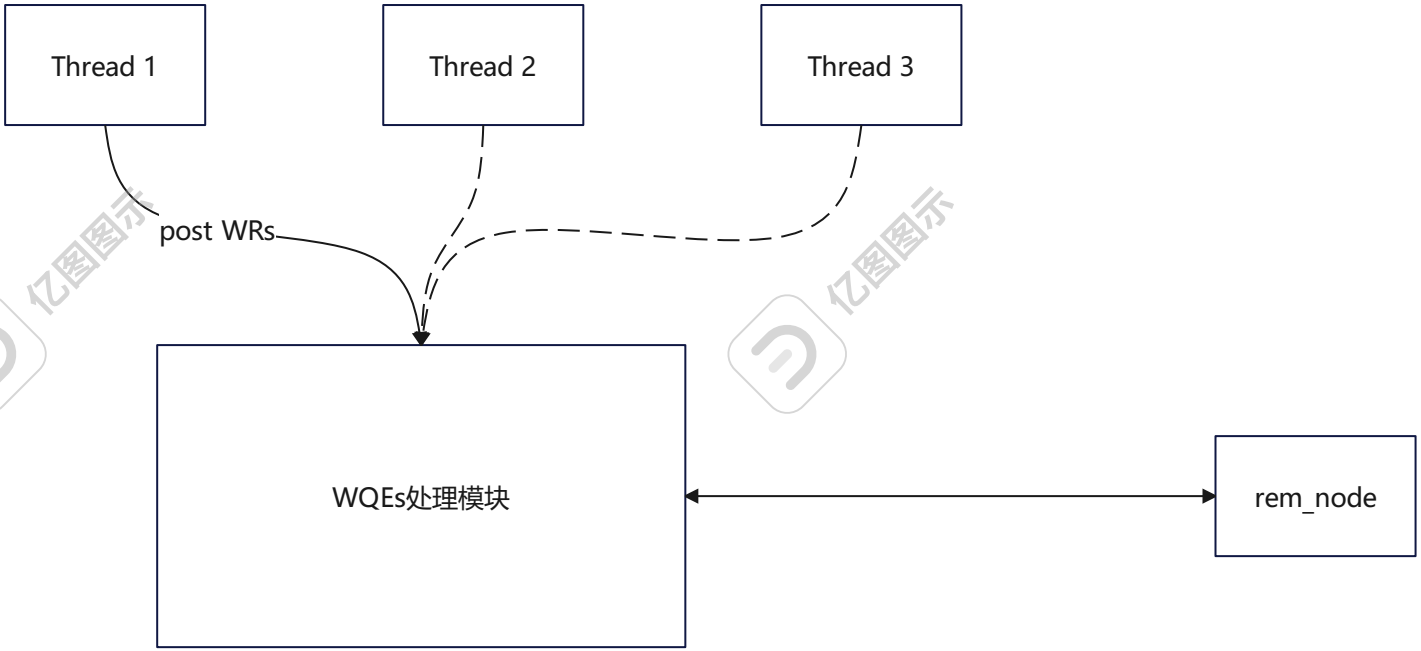
这里只讨论并展示
最基本的场景

与write等单端操作相比，此处多了server的post_recv操作，以及对应的poll_cq操作。进而统计出RR操作的带宽。
基于程序的优秀设计（**提前大量执行post_recv**，循环的时候cqe会在接收对应的SR后直接产生，可以直接poll cq, 无需等待），两端操作在QP connect之后就没有再进行握手，所以这个统计是很准确的，不存在程序设计不合理引入的等待、空转等问题。所以我们才能看到两个设备测出的结果有可能不一样。但是也应该比较接近，尤其是远程通信，因为中间经过的链路是一致的。



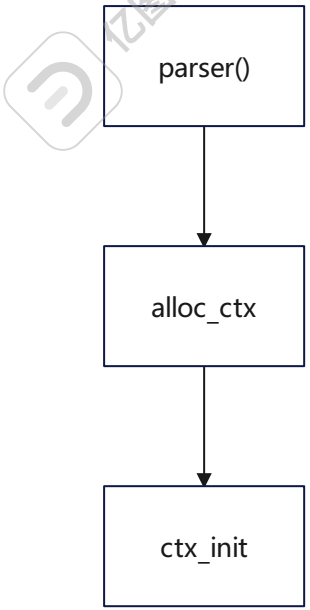

run_iter_lat_send
print_report_lat
LAT测试自动会在双端打印出结果，不需要-b参数；也不需要指定QP数量的-q参数。


run_iter_lat_send
print_report_lat
LAT测试自动会在双端打印出结果，不需要-b参数；也不需要指定QP数量的-q参数。



几个现象：
1.单线程操作，在单QP无法打满带宽时，单纯增大QP的数量无法提升带宽；
2.单线程带宽未打满时，多线程操作（通过执行多路perftest模拟）可以显著提升总带宽直至打满；
3.message size很大时，带宽是满的，因为要搬移数据，此时WQEs的处理速度比较慢，是瓶颈；
4.到远端再回来的时间间隔要比本地操作WR，处理WQEs的时间长的多；

所以提升带宽的方法主要是：
1.变小包为大包，传递message接近MTU，且有效数据占比大；
2.适当使用多线程，以压榨底层资源（多线程必然是使用多个QP）的工作密度；


parser()
alloc_ctx
ctx_init