

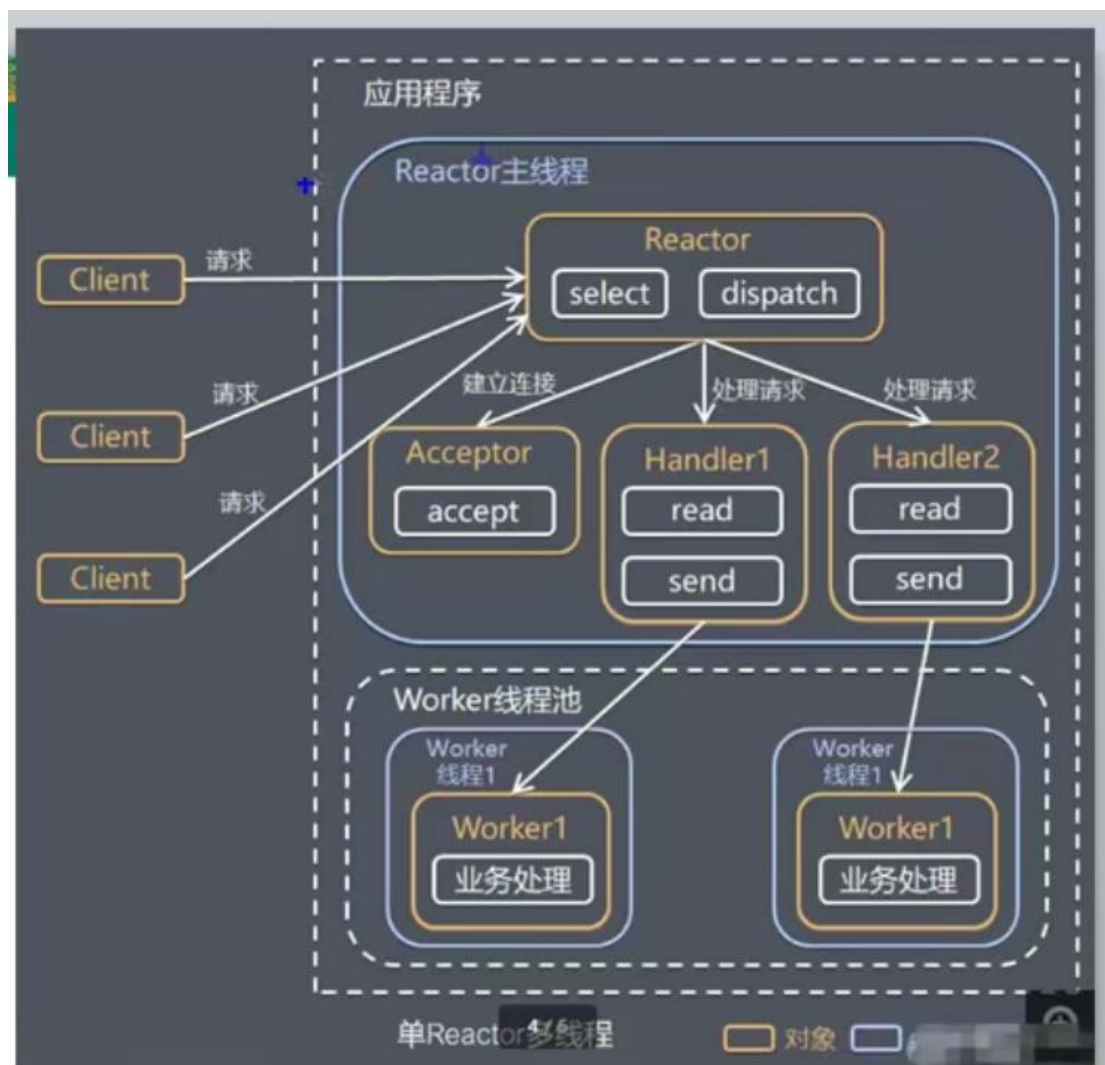
一、

- 1.Reactor 对象通过 select 监控客户端请求时间 收到事件后通过 dispatch 进行分发
- 2.如果建立连接请求，则由 Acceptor 通过 accept 处理连接请求，然后创建一个 Handler 对象处理完成连接后的各种事件
- 3.如果不是连接请求则由 Reactor 分发调用连接对应的 handler 来处理
- 4.handler 只负责响应事件，不做具体的业务处理，通过 read 读取数据后，会分发给后面的 worker 线程池的某个线程处理业务。
- 5.worker 线程池会分配独立线程完成真正的业务，并将结果返回给 handler，
- 6.handler 收到响应后通过 send 将结果返回给 client 客户端

优缺点：

优点：可以充分利用多核 cpu 处理能力

缺点：多线程数据共享和访问比较复杂,reactor 处理所有的事件的监听和响应，单线程运行，在高并发场景容易出现性能瓶颈。



二、主从 Reactor 多线程

针对单 Reactor 多线程模型中，Reactor 在单线程中运行，高并发场景下容易造成性能瓶颈，

可以让 Reactor 在多线程中运行

- 1.Reactor 主线程 MainReactor 对象通过 select 监听连接事件，收到事件后，通过 Acceptor 处理连接事件
- 2.当 Acceptor 处理连接事件后，MainReactor 将连接分配给 SubReactor
- 3.SubReactor 将连接加入到连接队列进行监听，并创建 handler 进行各种事件处理、
- 4.当有新事件发生时，SubReactor 就会调用对应的 handler 处理
- 5.handler 通过 read 读取数据，分发给后面的 worker 线程处理
- 6.worker 线程池分配独立的 worker 线程进行业务处理，并返回结果
- 7.handler 收到响应的结果后在通过 send 返回给 client
- 8.Reactor 主线程可以对应多个 Reactor 子线程,即 MainReactor 可以管理维护多个 SubReactor

优缺点:

优点:

- 1) 父线程和子线程的数据交互简单职责明确，父线程只需要接受新连接，子线程完成后续的业务处理
- 2) 父线程与子线程交互简单，Reactor 主线程只需要吧新连接传给子线程，子线程无需返回数据

缺点:

- 1) 编程复杂度较高

结合实例:

这种模型在许多项目中广泛使用，包括 Nginx 主从 Reactor 多进程模型，Memcached 主从多线程，Netty 主从多线程模型的支持

