线程的同步机制有哪些

互斥锁、信号量、自旋锁、条件变量、读写锁

互斥锁：为了确保同一时间只有一个线程访问数据，在访问共享资源前需要对互斥量上锁。一旦对互斥量上锁后，任何其他试图再次对互斥量上锁的线程都会被阻塞，即进入等待队列，当其他线程释放互斥量后，操作系统会激活那个被挂起的线程，让其投入运行。

信号量：信号量（sem）和互斥锁的区别：互斥锁只允许一个线程进入临界区，而信号量允许多个线程进入临界区

自旋锁：自旋锁与互斥量最不同的是：非阻塞锁，它不会被挂起，而是在获取锁之前一直处于忙等，即不停在消耗，执行循环。适用于多核处理器、临界区无阻塞情况，其中一个 CPU 上的线程进入临界区，另一个 CPU 上的线程尝试获取锁会自旋，因为它不会阻塞，所以只要稍稍等一下下就能进入临界区（预计线程等待锁的时间很短，短到比线程两次上下文切换时间要少的情况下），对于多核 CPU 来说，会提高并发率。

条件变量：互斥量不是万能的，比如某个线程正在等待共享数据内某个条件出现，可能需要重复对数据对象加锁和解锁（轮询），但是这样轮询非常耗费时间和资源，而且效率非常低，所以互斥锁不太适合这种情况

我们需要这样一种方法：当线程在等待满足某些条件时使线程进入睡眠状态，一旦条件满足，就唤醒线程

在条件变量的内部，有一次解锁加锁过程，条件不满足，将本线程加入等待队列，同时将传入的 mutex 变量解锁，一旦等待队列中的线程被唤醒，会再次对传入的 mutex 变量加锁！

读写锁：可以多个线程同时读，但是不能多个线程同时写，锁处于读模式时可以线程共享，而锁处于写模式时只能独占，所以读写锁又叫做共享-独占锁，读写锁比互斥锁更加具有适用性和并行性，最适用于对数据结构的读操作读操作次数多余写操作次数的场合。

线程池中的工作线程是一直等待吗

你的线程池工作线程处理完一个任务后的状态是什么？

进行等待，看请求队列中有无请求，无的就阻塞等待，有的话进行处理

如果同时1000个客户端进行访问请求，线程数不多，怎么能及时响应处理每一个呢？

因为这个项目采用的是proactor，主线程读取完数据才插入任务队列。这时候可以换成普通reactor，每个线程去接收数据，进一步的可以使用主从reactor。

如果一个客户请求需要占用线程很久的时间，会不会影响接下来的客户请求呢，有什么好的策略呢?

如果没有线程数限制，目前项目是固定线程数，可以改成动态线程池，连接请求过多的时候增加线程，对线程池进行一个扩充