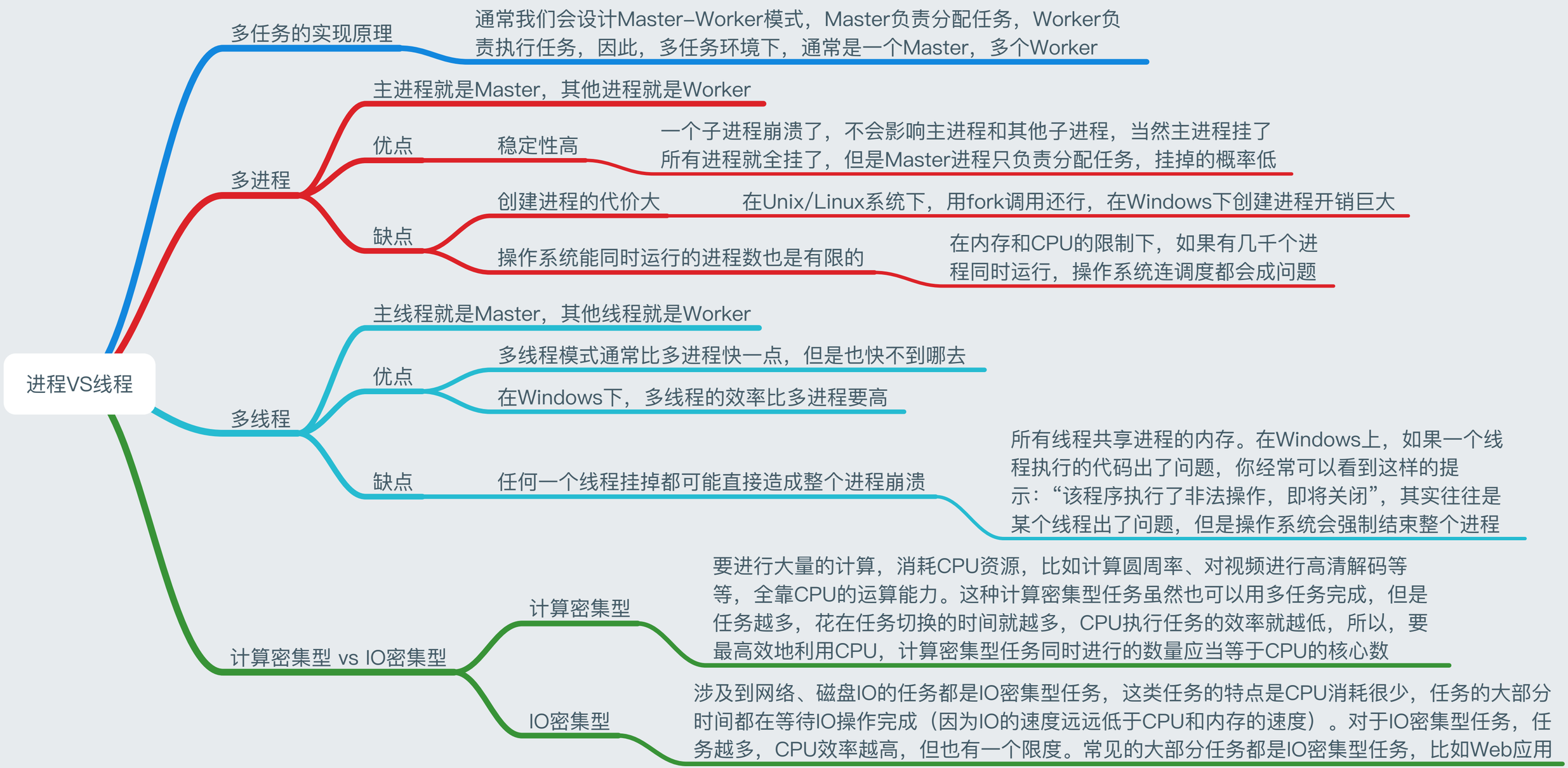
进程VS线程
多任务的实现原理
通常我们会设计Master-Worker模式，Master负责分配任务，Worker负责执行任务，因此，多任务环境下，通常是一个Master，多个Worker
多进程
主进程就是Master，其他进程就是Worker
优点
稳定性高
一个子进程崩溃了，不会影响主进程和其他子进程，当然主进程挂了所有进程就全挂了，但是Master进程只负责分配任务，挂掉的概率低
缺点
创建进程的代价大
在Unix/Linux系统下，用fork调用还行，在Windows下创建进程开销巨大
操作系统能同时运行的进程数也是有限的
在内存和CPU的限制下，如果有几千个进程同时运行，操作系统连调度都会成问题
多线程
主线程就是Master，其他线程就是Worker
优点
多线程模式通常比多进程快一点，但是也快不到哪去
在Windows下，多线程的效率比多进程要高
缺点
任何一个线程挂掉都可能直接造成整个进程崩溃
所有线程共享进程的内存。在Windows上，如果一个线程执行的代码出了问题，你经常可以看到这样的提示：“该程序执行了非法操作，即将关闭”，其实往往是某个线程出了问题，但是操作系统会强制结束整个进程
计算密集型 vs IO密集型
计算密集型
要进行大量的计算，消耗CPU资源，比如计算圆周率、对视频进行高清解码等等，全靠CPU的运算能力。这种计算密集型任务虽然也可以用多任务完成，但是任务越多，花在任务切换的时间就越多，CPU执行任务的效率就越低，所以，要最高效地利用CPU，计算密集型任务同时进行的数量应当等于CPU的核心数
IO密集型
涉及到网络、磁盘IO的任务都是IO密集型任务，这类任务的特点是CPU消耗很少，任务的大部分时间都在等待IO操作完成（因为IO的速度远远低于CPU和内存的速度）。对于IO密集型任务，任务越多，CPU效率越高，但也有一个限度。常见的大部分任务都是IO密集型任务，比如Web应用