# GIL

在非python环境中，单核情况下，同时只能有一个任务执行。多核时可以支持多个线程同时执行。但是在python中，无论有多少核，同时只能执行一个线程。究其原因，这就是由于GIL的存在导致的。

GIL的全称是Global Interpreter Lock(全局解释器锁)，来源是python设计之初的考虑，为了数据安全所做的决定。某个线程想要执行，必须先拿到GIL，我们可以把GIL看作是“通行证”，并且在一个python进程中，GIL只有一个。拿不到通行证的线程，就不允许进入CPU执行。GIL只在cpython中才有，因为cpython调用的是c语言的原生线程，所以他不能直接操作cpu，只能利用GIL保证同一时间只能有一个线程拿到数据。而在pypy和jpython中是没有GIL的。

Python多线程的工作过程：

python在使用多线程的时候，调用的是c语言的原生线程。

1. 拿到公共数据
2. 申请gil
3. python解释器调用os原生线程
4. os操作cpu执行运算
5. 当该线程执行时间到后，无论运算是否已经执行完，gil都被要求释放
6. 进而由其他进程重复上面的过程
7. 等其他进程执行完后，又会切换到之前的线程（从他记录的上下文继续执行）
8. 整个过程是每个线程执行自己的运算，当执行时间到就进行切换（context switch）。

python针对不同类型的代码执行效率也是不同的：参考博客：

http://python.jobbole.com/86822/

python下想要充分利用多核CPU，就用多进程。因为每个进程有各自独立的GIL，互不干扰，这样就可以真正意义上的并行执行，在python中，多进程的执行效率优于多线程(仅仅针对多核CPU而言)。

GIL在python中的版本差异：

1、在python2.x里，GIL的释放逻辑是当前线程遇见IO操作或者ticks计数达到100时进行释放。（ticks可以看作是python自身的一个计数器，专门做用于GIL，每次释放后归零，这个计数可以通过sys.setcheckinterval 来调整）。而每次释放GIL锁，线程进行锁竞争、切换线程，会消耗资源。并且由于GIL锁存在，python里一个进程永远只能同时执行一个线程(拿到GIL的线程才能执行)，这就是为什么在多核CPU上，python的多线程效率并不高。

2、在python3.x中，GIL不使用ticks计数，改为使用计时器（执行时间达到阈值后，当前线程释放GIL），这样对CPU密集型程序更加友好，但依然没有解决GIL导致的同一时间只能执行一个线程的问题，所以效率依然不尽如人意。