PYTHON GIL

MUXI WEB BACKEND SHARE

@NEO1218

GIL是什么鬼?

GIL是一把锁

GIL是一把大锁

GIL是一把全局解释锁

“什么是锁???”

—namewentilaile

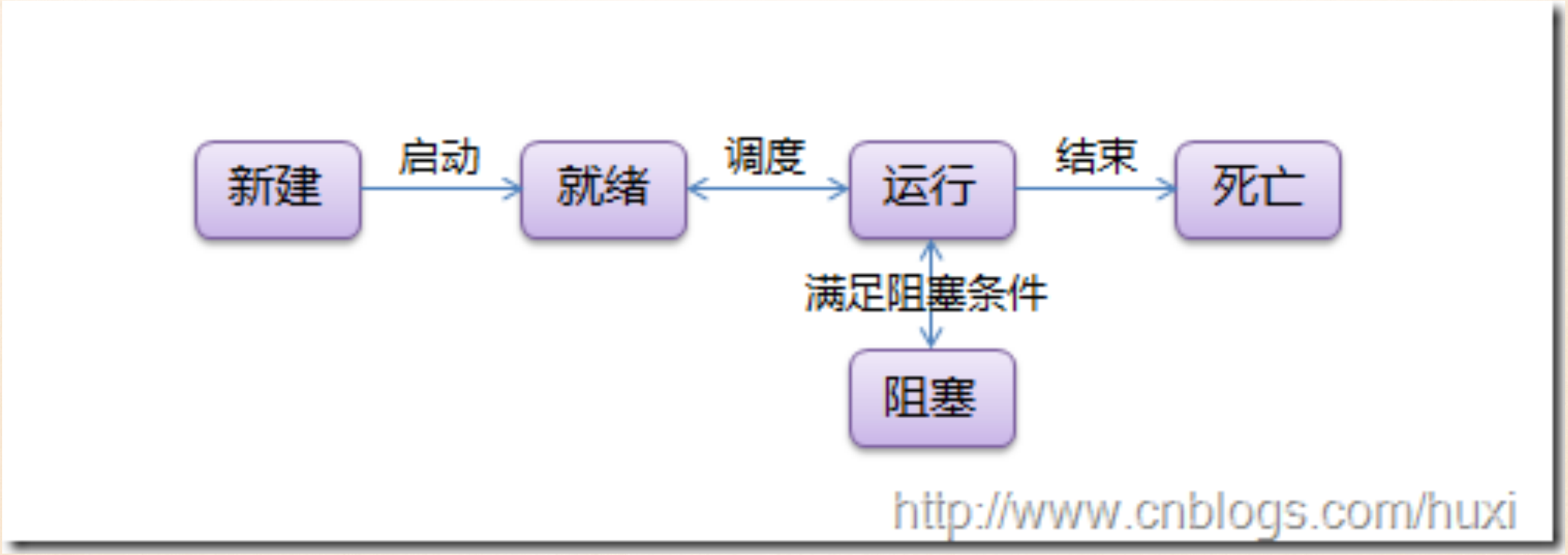
PYTHON线程编程

进程与线程

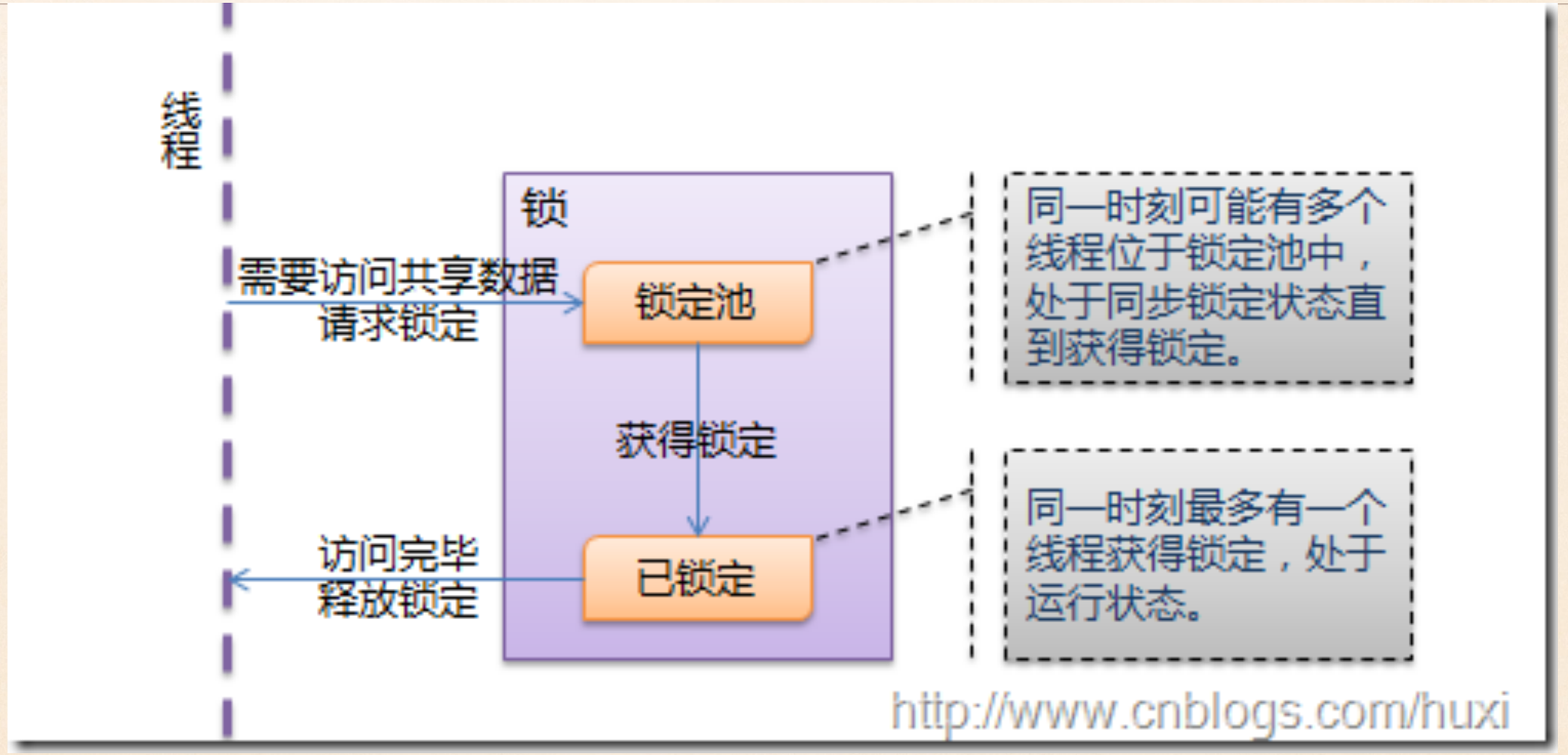
进程
进程是一种古老而典型的上下文系统，每个进程有独立的地址空间，资源句柄，他们互相之间不发生干扰。每个进程在内核中会有一个数据结构进行描述，我们称其为进程描述符。这些描述符包含了系统管理进程所需的信息，并且放在一个叫做任务队列的队列里面。

线程
线程是一种轻量进程，实际上在linux内核中，两者几乎没有差别，除了一点——线程并不产生新的地址空间和资源描述符表，而是复用父进程的。
但是无论如何，线程的调度和进程一样，必须陷入内核态。

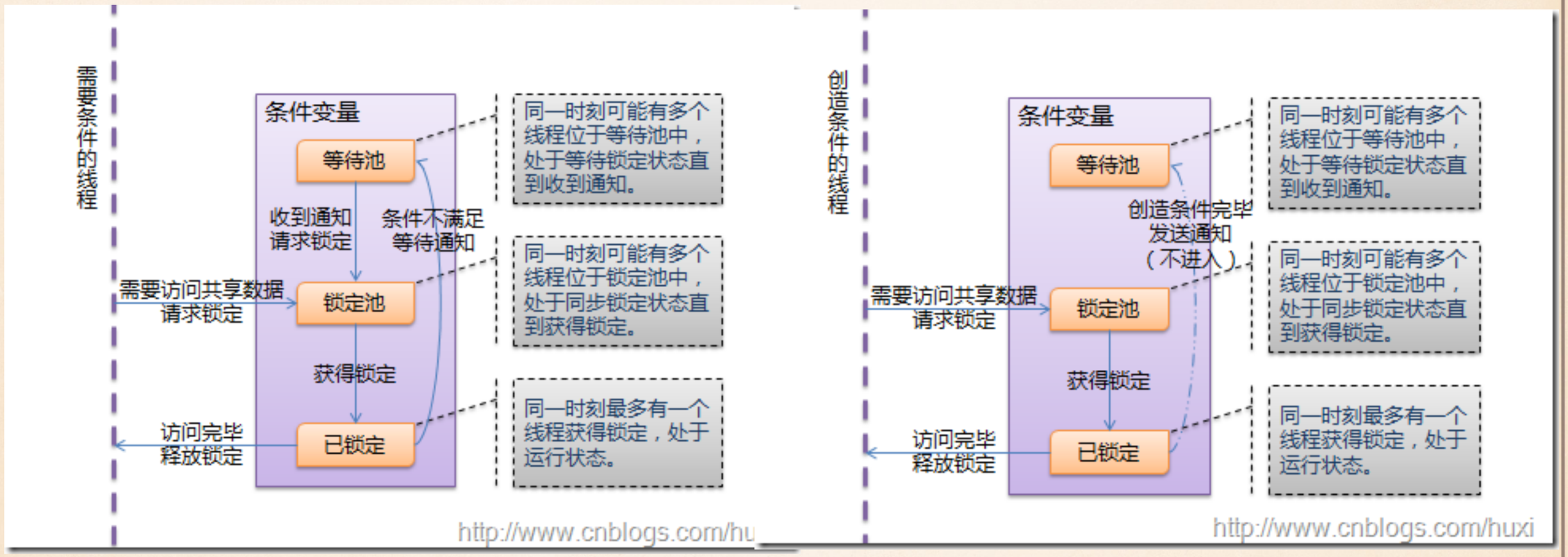
线程状态



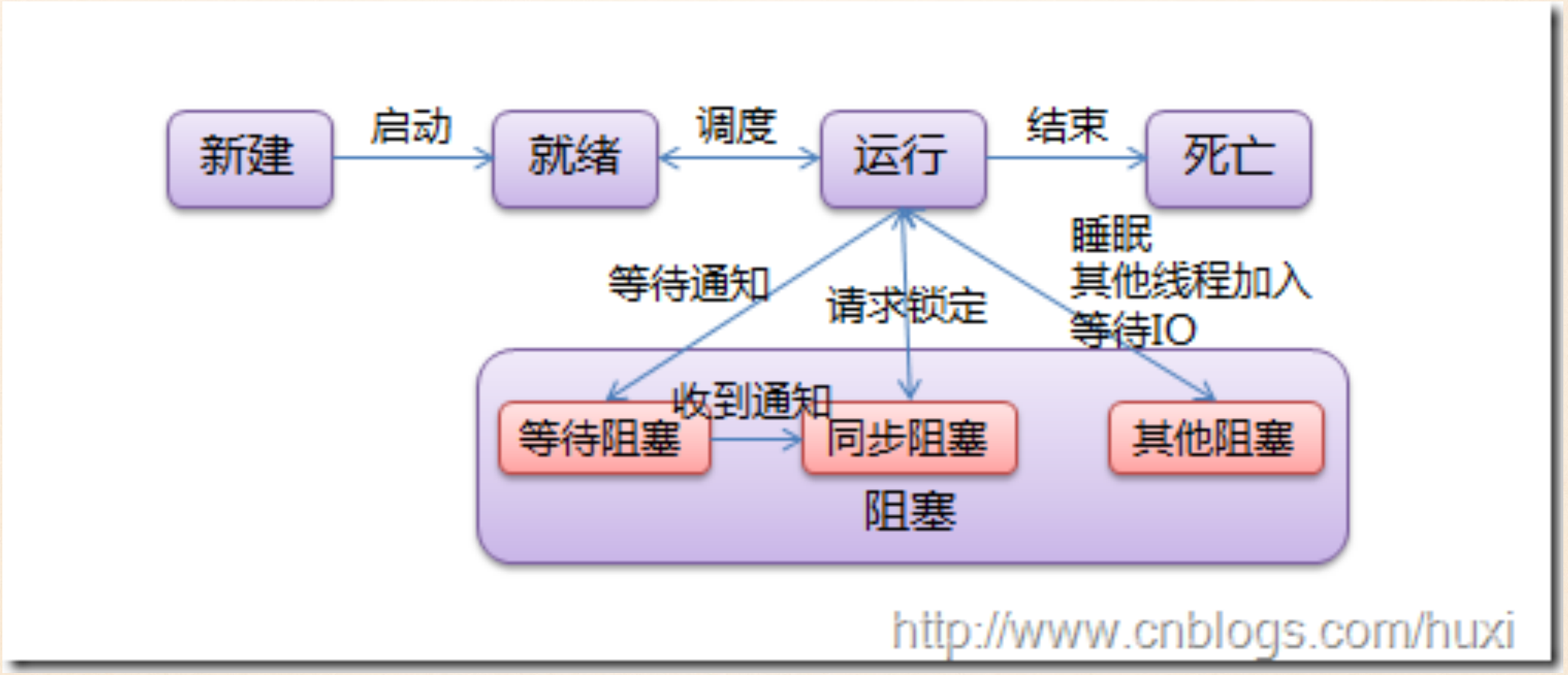
线程同步(锁)


线程
锁
需要访问共享数据
请求锁定
锁定池
获得锁定
访问完毕
释放锁定
已锁定
同一时刻可能有多个线程位于锁定池中，处于同步锁定状态直到获得锁定。
同一时刻最多有一个线程获得锁定，处于运行状态。
http://www.cnblogs.com/huxi

线程通信(条件变量)


需要条件的线程
条件变量
等待池
收到通知
请求锁定
条件不满足
等待通知
需要访问共享数据
请求锁定
锁定池
获得锁定
访问完毕
释放锁定
已锁定
同一时刻可能有多个线程位于等待池中，处于等待锁定状态直到收到通知。
同一时刻可能有多个线程位于锁定池中，处于同步锁定状态直到获得锁定。
同一时刻最多有一个线程获得锁定，处于运行状态。
http://www.cnblogs.com/hu
创造条件的线程
条件变量
等待池
创造条件完毕
发送通知
（不进入）
需要访问共享数据
请求锁定
锁定池
获得锁定
访问完毕
释放锁定
已锁定
同一时刻可能有多个线程位于等待池中，处于等待锁定状态直到收到通知。
同一时刻可能有多个线程位于锁定池中，处于同步锁定状态直到获得锁定。
同一时刻最多有一个线程获得锁定，处于运行状态。
http://www.cnblogs.com/huxi

线程运行与阻塞状态的转换



THREADING模块

PYTHON代码实例

生产者消费者模型

GIL其实不是PYTHON的菜

都是CPYTHON的锅


我好害怕

GIL的官方解释

In CPython, the global interpreter lock, or GIL, is a mutex that prevents multiple native threads from executing Python bytecodes at once. This lock is necessary mainly because CPython's memory management is not thread-safe. (However, since the GIL exists, other features have grown to depend on the guarantees that it enforces.)

为什么会有GIL

- 为了利用多核，Python开始支持多线程。而解决多线程之间数据完整性和状态同步的最简单方法自然就是加锁。 于是有了GIL这把超级大锁，而当越来越多的代码库开发者接受了这种设定后，他们开始大量依赖这种特性（即默认python内部对象是thread-safe的，无需在实现时考虑额外的内存锁和同步操作）。

GIL到底差在哪里?

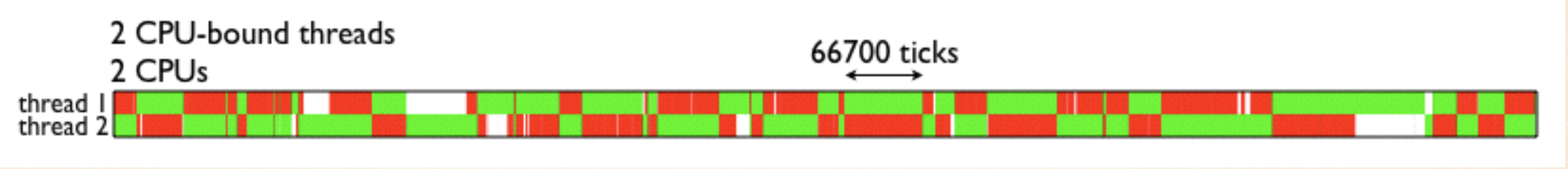
- 一个单线程和多线程程序测试

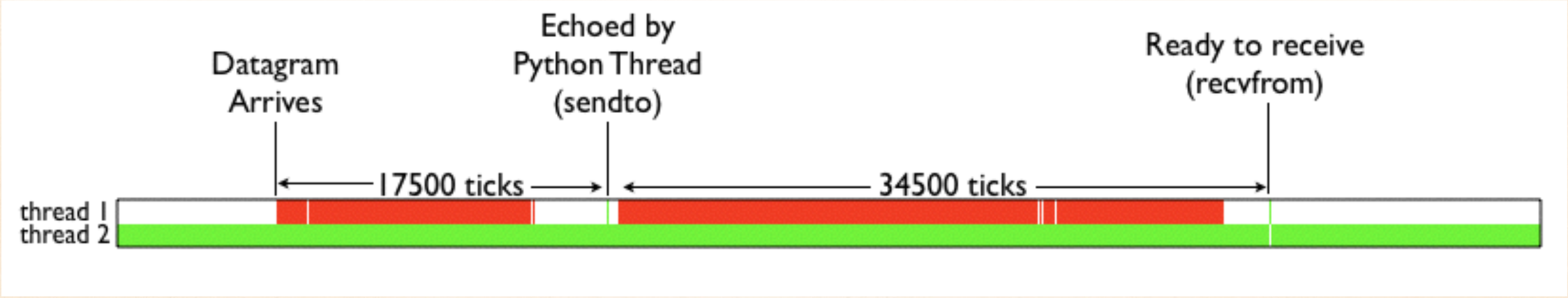
GIL设计上的缺陷

- 1. CPython本身没有实现线程调度，通过操作系统的调度算法进行调度。为了让各个线程能够平均利用CPU时间，python会计算当前已执行的微代码数量，达到一定阈值后就强制释放GIL。而这时也会触发一次操作系统的线程调度。

- 2.CPython中达到阈值释放GIL和唤醒线程获取GIL之间几乎是没有间隙的。

- 3. 问题是多核情况下, 在其他核心上的线程被唤醒时，大部分情况下主线程已经又再一次获取到GIL了。这个时候被唤醒执行的线程只能白白的浪费CPU时间，然后达到切换时间后进入待调度状态，再被唤醒，再等待，以此往复恶性循环。


2 CPU-bound threads
2 CPUs
66700 ticks
thread 1
thread 2


Datagram
Arrives
Echoed by
Python Thread
(sendto)
Ready to receive
(recvfrom)
17500 ticks
34500 ticks
thread 1
thread 2

如何避免?

- 1. 使用其他解释器
- 2. 多进程代替多线程
- 3. 交给刘聪了

全剧终&QA

谢谢大家