School of Economics ang Management, Beihang University

现代程序设计技术

赵吉昌
jichang@buaa.edu.cn

本周内容

- 面向对象编程
  - 多进程
  - 多线程

多进程

- 内存共享
  - 通过Value，Array实现内存共享
  - 通过Manager实现内存共享
  - Demo： mpm.py

多进程

- 进程池
  - 进程开启过多导致效率下降（同步、切换成本）
  - 应固定工作进程的数目
    - 由这些进程执行所有任务，而非开启更多的进程
    - 与CPU的核数相关

多进程

- 创建进程池
  - Pool([numprocess [,initializer [, initargs]]])
    - numprocess
      - 要创建的进程数，默认使用os.cpu_count()的值
    - initializer
      - 每个工作进程启动时要执行的可调用对象，默认为None
    - initargs
      - initializer的参数
  - p.apply()
    - 同步调用

多进程

- 进程池
  - p.apply_async()
    - 异步调用
    - 回调函数：进程池中任意任务完成后会立即通知主进程，主进程将调用另一个函数去处理该结果，该函数即回调函数,其参数为返回结果
  - p.map()
  - p.close()
    - 关闭进程池
  - p.join()
    - 等待所有工作进程退出，只能在close()或teminate()之后调用

多进程

- Demo
  - mppo.py

多进程

- ProcessPoolExecutor
  - 对multiprocessing进一步抽象
  - 提供更简单、统一的接口
  - submit(fn, *args, **kwargs)
    - returns a Future object representing the execution of the callable
  - map(func, *iterables, timeout=None)
    - func is executed asynchronously, i.e., several calls to func may be made concurrently and returns an iterator of results

多进程

- ProcessPoolExecutor
  - shutdown(wait=True)
    - signal the executor that it should free any resources that it is using when the currently pending futures are done executing
    - regardless of the value of wait, the entire Python program will not exit until all pending futures are done executing
    - can avoid having to call this method explicitly if you use the with statement
  - Demo: mpoe.py

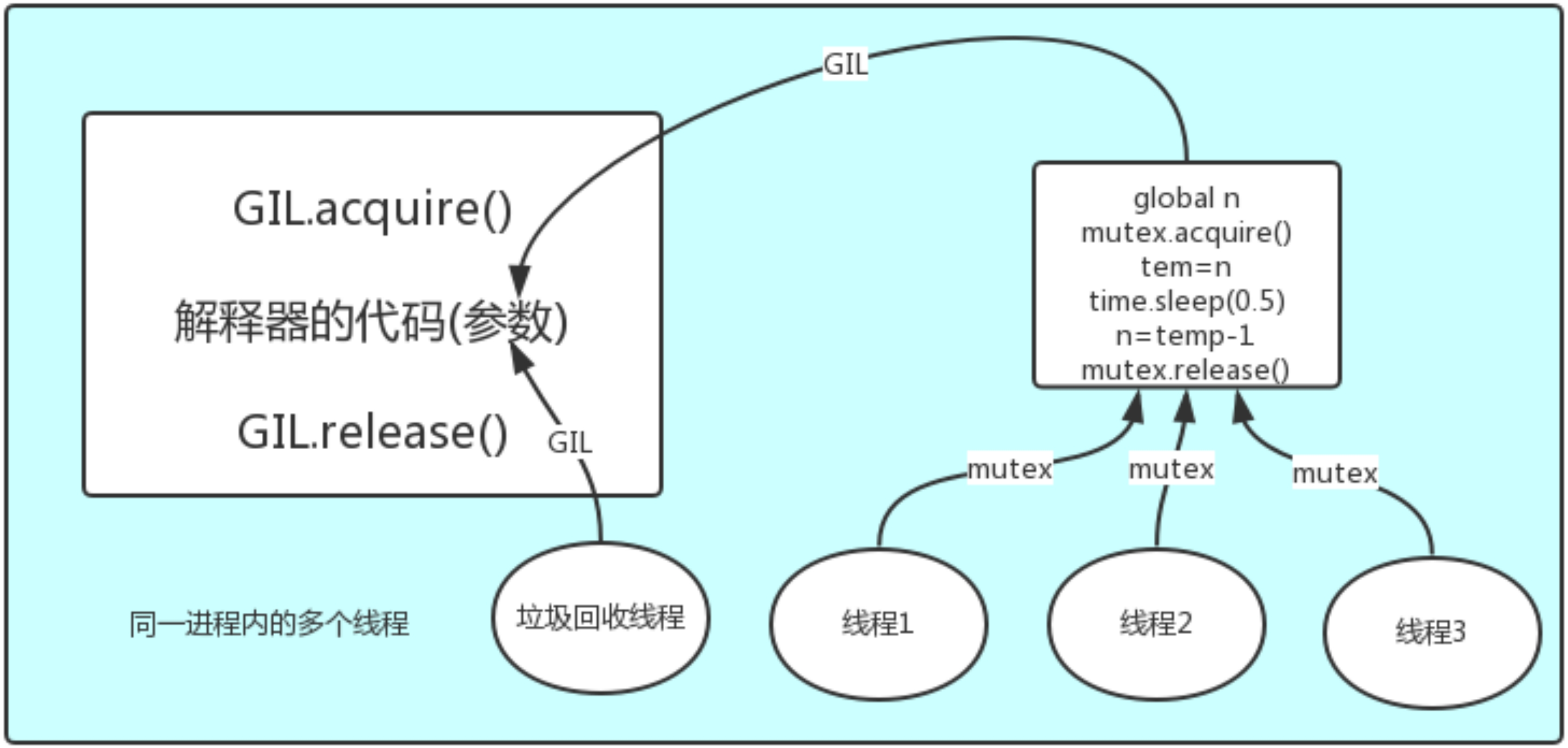
多进程

- 分布式多进程
  - 多机环境
  - 跨设备数据交换
  - 如master-worker模型
  - 通过manager暴露Queue
  - Demo: mpd_server.py, mpd_worker.py mp_qm.py

多线程

- GIL(Global Interpreter Lock)
  - GIL非Python特性，而是实现Python解释器（Cpython）时引入的概念
  - GIL本质上是互斥锁，控制同一时间共享数据只能被一个任务修改，以保证数据安全
  - GIL在解释器级保护共享数据，在用户编程层面保护数据则需要自行加锁处理
  - Cpython解释器中，同一个进程下开启的多线程，同一时刻只能有一个线程执行，无法利用多核优势
    - 可能需要先获取GIL

多进程


GIL
GIL.acquire()
解释器的代码(参数)
GIL.release()
GIL
global n
mutex.acquire()
tem=n
time.sleep(0.5)
n=temp-1
mutex.release()
mutex
mutex
mutex
同一进程内的多个线程
垃圾回收线程
线程1
线程2
线程3

多线程

- GIL
  - 进程可以利用多核，但是开销大，而多线程开销小，但却无法利用多核
    - If you want your application to make better use of the computational resources of multi-core machines, you are advised to use multiprocessing or concurrent.futures.ProcessPoolExecutor
    - 多进程用于计算密集型，如金融分析
    - However, threading is still an appropriate model if you want to run multiple I/O-bound tasks simultaneously.
    - 多线程用于IO密集型，如socket，爬虫，web

多线程

- 多线程编程
  - threading模块
  - multiprocess模块和threading模块在使用层面十分相似
  - threading.currentThread()
    - 返回当前的线程实例
  - threading.enumerate()
    - 返回所有正在运行线程的list
  - threading.activeCount()
    - 返回正在运行的线程数量，与len(threading.enumerate())结果相同

多线程

- 创建多线程
  - 通过指定target参数
  - 通过继承Thread类
  - 设置守护线程
    - setDaemon(True)
    - 应在start()之前
  - Demo: mt.py

多线程

- 线程同步
  - 锁(threading.Lock, threading.RLock,可重入锁)
  - 信号量(threading.Semaphore)
  - 事件(threading.Event)
  - 条件(threading.Condition)
    - Demo: mtc.py
  - 定时器(threading.Timer)
    - Demo: mtt.py

多线程

- 线程同步
  - Barrier
    - This class provides a simple synchronization primitive for use by a fixed number of threads that need to wait for each other.
    - Each of the threads tries to pass the barrier by calling the wait() method and will block until all of the threads have made their wait() calls. At this point, the threads are released simultaneously.
    - Demo: mtb.py

多线程

- 队列
  - queue.Queue
  - queue.LifoQueue
  - queue.PriorityQueue （元素是元组，第一个元素为优先级）
  - Demo: mtq.py

多线程

- 线程池
  - ThreadPoolExecutor
    - assuming that ThreadPoolExecutor is often used to overlap I/O instead of CPU work and the number of workers should be higher than the number of workers for ProcessPoolExecutor
    - `as_completed(fs, timeout=None)`
      - Returns an iterator over the Future instances given by fs that yields futures as they complete (finished or cancelled futures)
    - Demo: mtp.py

多进程和多线程的比较

- Demo: ptc.py