**Pyhton 如何实现多线程并行计算**

1. **串行计算**

先看一段简单的代码：

import time

t0=time.time()

for k in range(30):

values=k\*k

print(values)

time.sleep(1)

t1=time.time()

print('运行时间为:',int(t1-t0))

上面的例子中，我们用一个 **for** 循环计算自然数的平方。这里我们一个自然数计算完才能接着计算另外一个数。这种计算方式我们称为“串行计算”。早期为什么采用这种串行计算呢？因为以前个人电脑CPU是单核的，硬件的条件决定了程序的处理方式。

能不能几个数同时计算？好比如在银行的营业厅排队，如果只开一个窗口办理业务，你需要等前面一个人办完，才轮到你，如果能开多个窗口，显然会快很多。这种开多个窗口处理业务的想法，在计算机中的应用就是“并行计算”。多个窗口对应的就是计算机有多个核。(理解了“并行计算”，就容易进一步理解分布式计算。）

1. **多核与线程**

个人电脑的处理器最早是单核的。

多内核（multicore chips）是指在一枚处理器（chip）中集成两个或多个完整的计算引擎(内核)。

2005年4月，英特尔仓促推出简单封装双核的奔腾D和奔腾四至尊版840。

但真正的“双核元年”，则被认为是2006年。这一年的7月23日，英特尔基于酷睿(Core)架构的处理器正式发布。2006年11月，又推出面向服务器、工作站和高端个人电脑的至强(Xeon)5300和酷睿双核和四核至尊版系列处理器。

进入2009年，处理器已经由双核升级到四核时代，在斯坦福大学召开的Hot Chips大会上，IBM、富士通、AMD和Intel等众多芯片制造商展示出其六核、八核等多核服务器处理器出现。

线程，有时被称为轻量级进程(Lightweight Process，LWP），是程序执行流的最小单元。一个标准的线程由线程ID，当前指令指针(PC），寄存器集合和堆栈组成。另外，线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一点儿在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。每一个程序都至少有一个线程，若程序只有一个线程，那就是程序本身。

线程是程序中一个单一的顺序控制流程。进程内一个相对独立的、可调度的执行单元，是系统独立调度和分派CPU的基本单位指运行中的程序的调度单位。在单个程序中同时运行多个线程完成不同的工作，称为**多线程。**

1. **Python并行计算模块**

现在个人电脑都是多核多线程，比如[Intel Core i5-6200U](http://product.pconline.com.cn/notebook_cpu/intel/598596.html" \o "Intel Core i5-6200U)是双核四线程。如果我们的程序采用串行计算，那么就有一个内核是闲置的。我们如何把剩余的电脑资源充分利用起来，提高电脑的处理效率。我们可以采用多线程并行计算的方式来利用多核资源。

Python标准库为我们提供了threading和multiprocessing模块编写相应的多线程/多进程代码。从Python3.2开始，标准库为我们提供了concurrent.futures模块，它提供了ThreadPoolExecutor和ProcessPoolExecutor两个类，实现了对threading和multiprocessing的更高级的抽象，对编写线程池/进程池提供了直接的支持。 下面我们有一个测试案例，来看看如何用concurrent.futures实现并行计算。

1. **案例与测试**

**4.1 先运行串行计算代码**

import time

t0=time.time()

for k in range(30):

values=k\*k

print(values)

time.sleep(1)

t1=time.time()

print('运行时间为:',int(t1-t0))

输出结果：

0

1

4

9

16

25

.....................

729

784

841

**运行时间为: 30**

**4.2 并行计算代码**

from concurrent.futures import ThreadPoolExecutor as Pool

import time

s0=time.time()

def sqare(x):

time.sleep(1)

return x\*x

number\_list=range(30)

pool = Pool(max\_workers=5) #设置最大线程为5

result=pool.map(sqare,number\_list) #map并行计算

for k in result:

print(k)

s1=time.time()

print('运行时间为:',int(s1-s0))

输出结果：

0

1

4

..........

400

676

729

784

841

**运行时间为: 6**