

# 并行计算作业补充（Python实现）

course

---

## 前情提要

之前由于并行计算的作业被视为类同或抄袭网络，正好最近学习Python，于是刚好在这里作一个补充，新方法采用Python实现并行，一来学习用，二来为了完成作业。

旧作业地址如下：

[OpenMP计算圆周率](#)

## Python并行计算

Python在并行计算方面使用的是GIL(Global Interpreter Lock，全局解释器锁)，被认为的多线程其实是伪的，比较鸡肋，但在实验环境中，配合GPU，其实还不是很鸡肋，在此我们就讲解一下；

Python的原解释器CPython是有GIL的，在执行代码过程中，会产生互斥锁来限制线程对共享资源的访问，而GIL的作用就是，一个进行同一时间只能允许一个线程运算，摆明的单线程啊！

由于CPython中的GIL的存在我们可以暂时不奢望能在CPython中使用多线程利用多核资源进行并行计算了，因此我们在Python中可以利用多进程的方式充分利用多核资源。

并行计算的目的是将所有的核心都运行起来以提高代码的执行速度，在python中由于存在全局解释器锁（GIL）如果使用默认的python多线程进行并行计算可能会发现代码的执行速度并不会加快，甚至会比使用单核心要慢！！！！

## 用到的库

下面就来简单介绍几个Python并行计算中用到的库，最后再使用Python实现一个小例子，应该可以完成作业了吧？

在Thread和Process中，应当优选Process，因为Process更稳定，而且，Process可以分布到多台机器上，而Thread最多只能分布到同一台机器的多个CPU上。当然我们处理一个简单的例子用哪个都可以。

这里我们使用Python的multiprocessing模块，其中managers子模块还支持把多进程分布到多台机器上。一个服务进程可以作为调度者，将任务分布到其他多个进程中，依靠网络通信。由于managers模块封装很好，不必了解网络通信的细节，就可以很容易地编写分布式多进程程序。

- 简单介绍一下multiprocessing模块的用法：

multiprocessing模块提供了一个Process类来代表一个进程对象。创建子进程时，只需要传入一个执行函数和函数的参数，创建一个Process实例，start()方法启动，join()方法可以等待子进程结束后再继续往下运行，通常用于进程间的同步。

代码举例（代码经过实验，在Python3上运行）：

```
1.  # -*- coding=utf-8 -*-
2.
3.  from multiprocessing import Process
4.  import os
5.  import time
6.
7.
8.  def run_process(name):
9.      time.sleep(2)
10.     print('Run child process %s is (%s). ' % (name, os.getpid()))
11.
12.     pass
13.
14. def hello_world():
15.     time.sleep(5)
16.     print('Run child process is (%s). ' % (os.getpid()))
17.
18.     pass
19.
20. if __name__ == "__main__":
```

```

21.
22.     print("Parent process is %s." % (os.getpid()))
23.
24.     p1 = Process(target=run_process, args=('test', ))
25.     p2 = Process(target=hello_world)
26.
27.     print(" process will start..... ")
28.
29.     p1.start()
30.     p2.start()
31.     p1.join()
32.
33.     print(" process end!")

```

输出如下：

```

Parent process is 85484.
 process will start.....
Run child process test is (85485).
 process end!
Run child process is (85486).
[Finished in 5.2s]

```

一些并行模块通过修改python的GIL机制突破了这个限制，使得Python在多核电脑中也能够有效的进行并行计算。PP ( Parallel Python ) 模块就是其中一种。

- PP ( Parallel Python ) 模块举例：

安装pp模块的时候出现了一个问题，在这里做出解释：

```

1.     pip install pp==1.6.5
2.     Collecting pp==1.6.5
3.         Using cached
         https://files.pythonhosted.org/packages/14/e9/f69030681985226849becd36b04
         e2c0cb99babff23c8342bc4e30ded06b2/pp-1.6.5.tar.gz
4.         Complete output from command python setup.py egg_info:
5.         Traceback (most recent call last):
6.             File "<string>", line 1, in <module>
7.             File
         "/private/var/folders/gk/4tnnvlmj0zzg74xzmknq8g0c0000gn/T/pip-install-
         1pnu5md4/pp/setup.py", line 12, in <module>
8.                 from pp import version as VERSION

```

```

9.         File
10.         "/private/var/folders/gk/4tnnvlmj0zzg74xzmknq8g0c0000gn/T/pip-install-
11.         lpnu5md4/pp/pp.py", line 121
12.             print sout,
13.             ^
14.         SyntaxError: Missing parentheses in call to 'print'. Did you mean p
rint(print sout, end=" ")?
15.
16. Command "python setup.py egg_info" failed with error code 1 in
17. /private/var/folders/gk/4tnnvlmj0zzg74xzmknq8g0c0000gn/T/pip-install-xa
18. 12jeey/pp/

```

具体原因是因为pip工具默认调用了python3的语法运行了setup.py脚本，但是该下载脚本默认用的貌似是python2的语法，解决方法如下：

```

1. pip install setuptools --upgrade --user

```

接着运行：

```

1. pip install pp

```

出现：

```

1. Collecting pp
2.   Using cached
   https://files.pythonhosted.org/packages/14/e9/f69030681985226849becd36b04
   e2c0cb99babff23c8342bc4e30ded06b2/pp-1.6.5.tar.gz
3.   Complete output from command python setup.py egg_info:
4.   Traceback (most recent call last):
5.     File "<string>", line 1, in <module>
6.     File
7.     "/private/var/folders/gk/4tnnvlmj0zzg74xzmknq8g0c0000gn/T/pip-install-
8.     i2brisl3/pp/setup.py", line 12, in <module>
9.         from pp import version as VERSION
10.     File
11.     "/private/var/folders/gk/4tnnvlmj0zzg74xzmknq8g0c0000gn/T/pip-install-
12.     i2brisl3/pp/pp.py", line 121
13.         print sout,
14.         ^
15.     SyntaxError: Missing parentheses in call to 'print'. Did you mean p
rint(print sout, end=" ")?

```

```

12.
13. -----
14. Command "python setup.py egg_info" failed with error code 1 in
    /private/var/folders/gk/4tnnvlmj0zzg74xzmknq8g0c0000gn/T/pip-install-i2
    brisl3/pp/

```

上述原因是使用pip ( python3版本 ) 工具时调用了python2的脚本，导致无法正常调用对应版本的setup.py脚本；

使用pip2安装出现：

```

1. Could not fetch URL https://pypi.python.org/simple/pp/: There was a p
    roblem confirming the ssl certificate: [SSL:
    TLSV1_ALERT_PROTOCOL_VERSION] tlsv1 alert protocol version
    (_ssl.c:590) - skipping
2. Could not find a version that satisfies the requirement pp (from ver
    sions: )
3. No matching distribution found for pp

```

所以可能是python3上的pp模块并不是这个名字，我们使用其它方法把这个模块下载下来：

```

1. wget
    https://files.pythonhosted.org/packages/14/e9/f69030681985226849becd36b04
    e2c0cb99babff23c8342bc4e30ded06b2/pp-1.6.5.tar.gz
2. tar xvfz pp-1.6.5
3. cd pp-1.6.5
4. python setup.py install

```

这样我们使用python2.7对pp模块进行了安装，而且pp模块的最新版本1.6.5是只支持python2的；接下来我们在python2.7下对pp模块进行引用，来完成并行计算的例子；

最后发现个问题，python版本对应有特定的pp版本，并不是不的支持了，详细见网站[PP模块版本](#)，但是环境已经配好那就用python2吧暂时；

- 例子举例，该例子实现了用并行计算的方法解决“从0到给定范围内所有质数的和”：

```

1. #-*-coding=utf-8-*-
2.
3. import time
4. import math

```

```

5.
6. def isprime(n):
7.     if not isinstance(n, int):
8.         raise TypeError("argument passed to is_prime is not of 'int'
type")
9.     if n < 2:
10.        return False
11.    if n == 2:
12.        return True
13.    max = int(math.ceil(math.sqrt(n)))
14.    i = 2
15.    while i <= max:
16.        if n % i == 0:
17.            return False
18.        i += 1
19.    return True
20.
21. def sum_primes(n):
22.     return sum([x for x in range(2, n) if isprime(x)])
23.
24.
25. #串行代码
26. print("{beg} serial process {beg}".format(beg='-'*16))
27. startTime = time.time()
28.
29. inputs = (100000, 100100, 100200, 100300, 100400, 100500, 100600,
100700)
30. results = [(input, sum_primes(input)) for input in inputs]
31.
32. for input, result in results:
33.     print("Sum of primes below %s is %s" % (input, result))
34.
35. print("use: %.3fs"%( time.time()-startTime))
36.
37.
38. import pp
39.
40. #并行代码
41. print("{beg} parallel process {beg}".format(beg='-'*16))
42. startTime = time.time()
43.
44. job_server = pp.Server()
45.
46. inputs = (100000, 100100, 100200, 100300, 100400, 100500, 100600,
100700)

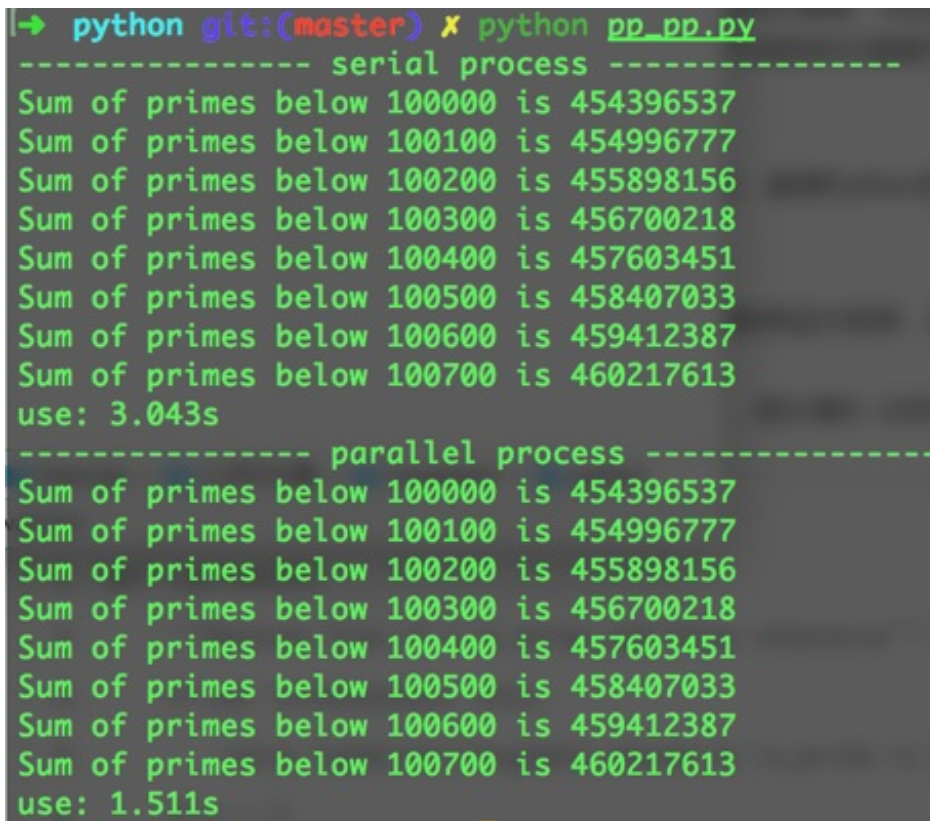
```

```

47. jobs = [(input, job_server.submit(sum_primes, (input, ), (isprime, ),
48.                                     ("math", )) ) for input in inputs]
49.
50. for input, job in jobs:
51.     print("Sum of primes below %s is %s" % (input, job()))
52.
53. print("use: %.3fs"%( time.time()-startTime ) )

```

运行得出结果：



```

python git:(master) x python pp_pp.py
----- serial process -----
Sum of primes below 100000 is 454396537
Sum of primes below 100100 is 454996777
Sum of primes below 100200 is 455898156
Sum of primes below 100300 is 456700218
Sum of primes below 100400 is 457603451
Sum of primes below 100500 is 458407033
Sum of primes below 100600 is 459412387
Sum of primes below 100700 is 460217613
use: 3.043s
----- parallel process -----
Sum of primes below 100000 is 454396537
Sum of primes below 100100 is 454996777
Sum of primes below 100200 is 455898156
Sum of primes below 100300 is 456700218
Sum of primes below 100400 is 457603451
Sum of primes below 100500 is 458407033
Sum of primes below 100600 is 459412387
Sum of primes below 100700 is 460217613
use: 1.511s

```

## 实验总结

这次作业补充让我学习了两个python并行计算的模块，一个是使用多线程方法的mutlprocessing，另一个是真正实现并行计算的pp模块，在今后的学习中，可以利用这些实现的并行计算去加速运行大规模的计算量，另外在深度学习方面我们有强大的numpy以及利用gpu的库，总的来说在并行计算的方面我们还有很多东西可以实践，今天就到这了，希望这个作业可以过。

