python 分布式进程模型

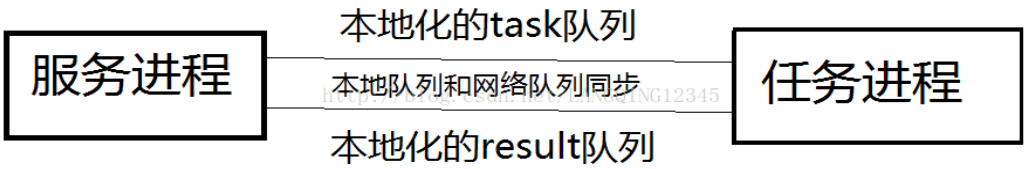
原创 2015 年 05 月 20 日 00:53:11

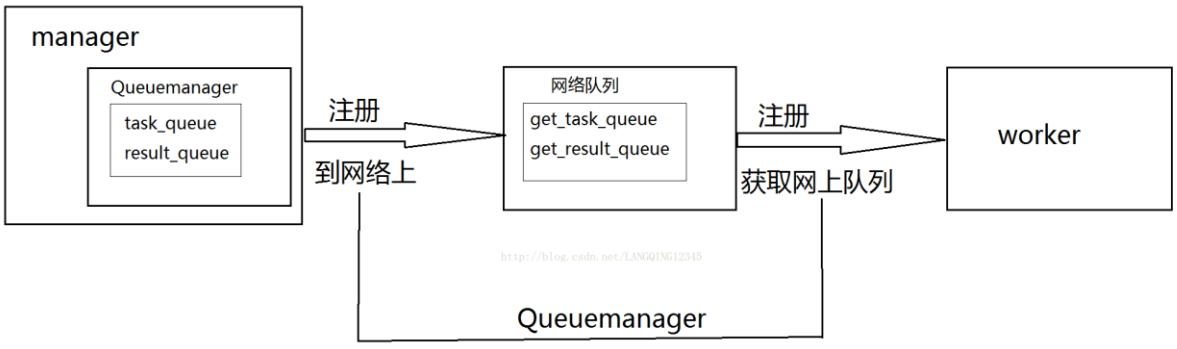
- 1349

分布式进程：Process 进程分布到多台机器上

Python 的 multiprocessing 模块不但支持多进程，其中 managers 子模块还支持把多进程分布到多台机器上。

可以写一个服务进程作为调度者，将任务分布到其他多个进程中，依靠网络通信进行管理。





当然，具体的任务操作还要网络队列本地化后才能进行，而本地化依赖于管理实例的创建。

另外值得注意的是，manager 和 worker 两端创建的实例是相关的，即 worker 的实例，通过验证码和网络 IP 地址连接到 manager 即服务器的实例。

```
[python] view plain copy print?
# -*- coding:utf-8 -*-

#taskmanager.py
'''''
1.建立队列，作为共享消息的通道。 服务进程
任务队列 task_queue 作为服务进程传递任务给任务进程的通道，而结果队列 result_queue 作为任务进程完成任务后回传给服务进程的通道。
值得注意的是，在【一台机器上】写多进程程序时，创建的 Queue 可以直接拿来用
然而，在分布式多进程环境下，不可以直接添加任务到原始的 task_queue，那样就绕过了 Queuemanager 的封装，
必须通过 manager.get_task_queue()获得的【Queue 接口】添加任务。
2.把 1.中建立的队列在网络上注册，暴露给其他进程(主机)，注册后获得【网络队列】(可以认为是 1.中队列的'映像')
3.建立一个对象(Queuemanager(BaseManager))实例 manager，绑定端口和验证码
4.启动 3.中建立的实例，以便监听连接(启动管理 manager，监管信息通道)
5.通过管理实例的方法获得通过网络访问的 Queue 对象，即再把网络队列实体化成可以使用的本地队列(通过本地上传到网络)
6.创建任务到“本地”队列中，自动上传任务到网络队列中，以供分配给
'''

import random,time,Queue
from multiprocessing.managers import BaseManager

#建立两个队列，分别存放任务和结果，它们用来进行进程间通信，交换对象。换言之，这两个队列就是交换对象
task_queue=Queue.Queue()
result_queue=Queue.Queue()

class Queuemanager(BaseManager):
    pass

#把创建的两个队列注册在网络上，利用 register 方法，callable 参数关联了 Queue 对象
```

```
#typeid is a “type identifier”(类型标识符) which is used to identify a particular type of shared object. This must be a string.
#callable is a callable used for creating objects for this type identifier.—后者用来创建前者，后者是具体的对象，而前者是利用后者创造出来的“影子”
Queuemanager.register('get_task_queue',callable=lambda:task_queue)
Queuemanager.register('get_result_queue',callable=lambda:result_queue)

#绑定端口 5000，设置验证码‘abc’。这个相当于对象的初始化
#address is the address on which the manager process listens for new connections
manager=Queuemanager(address=('',5000),authkey='abc')

#启动管理
manager.get_server().serve_forever()

#通过管理实例的方法获得通过网络访问的 Queue 对象
task=manager.get_task_queue()
result=manager.get_result_queue()

#放几个任务进去
for i in range(10):
    n=random.randint(0,10000)
    print 'put task %d ...' %n
    task.put(n)   #task 是本地队列

print 'try get result...'
for i in range(10):
    print 'result is %s' %result.get(timeout=10)

#关闭管理
manager.shutdown()
```

[python] view plain copy print?

```
# taskworker.py

import time, sys, Queue
from multiprocessing.managers import BaseManager
```

```
5.
6.  # 创建类似的 QueueManager:
7.  class QueueManager(BaseManager):
8.      pass
9.
10. # 由于这个 QueueManager 只从网络上获取 Queue，所以注册时只提供名字:
11. QueueManager.register('get_task_queue')
12. QueueManager.register('get_result_queue')
13.
14. # 连接到服务器，也就是运行 taskmanager.py 的机器:
15. server_addr = '127.0.0.1'
16. print('Connect to server %s...' % server_addr)
17. # 端口和验证码注意保持与 taskmanager.py 设置的完全一致:
18. m = QueueManager(address=(server_addr, 5000), authkey='abc')
19. # 从网络连接:
20. m.connect()
21. # 获取 Queue 的对象:
22. task = m.get_task_queue()
23. result = m.get_result_queue()
24. # 从 task 队列取任务,并把结果写入 result 队列:
25. for i in range(10):
26.     try:
27.         n = task.get(timeout=1)
28.         print('run task %d * %d...' % (n, n))
29.         r = '%d * %d = %d' % (n, n, n*n)
30.         time.sleep(1)
31.         result.put(r)
32.     except Queue.Empty:
33.         print('task queue is empty.')
34. # 处理结束:
35. print('worker exit.')
```

参考文档：

https://docs.python.org/2/library/multiprocessing.html?highlight=base manager#multiprocessing.sharedctypes.multiprocessing.Manager