Python 线程池使用有限大小的工作队列

2020-07-23 — Yanbin

在去年的一篇 Python 多线程编程 中学习了 Python 中如何使用多线程来调度任务，工作中也不时从自己的博客中找来参考。在运用当中不时的碰到内存消耗殆尽情况，直接把命令行窗口打死，不得不强行关窗口或杀进程。之前一直未意识到问题所在，只知任务太多就必死无疑，现在要用 Python 来处理大量任务了，必须着手来解决一下它。其实原因很简单，和 Java 的 ThreadPoolExecutor 一样(看它们用的类名都是一样的)。Java 的 ThreadPoolExecutor 内部使用了一个 `Integer.MAX_VALUE` 的 LinkedBlockingQueue 来存放提交的待处理的任务，所以基本上就是一个无底洞，自然解决办法也是类似的，需要一个 Bounded Queue 来存放任务列表。

在解决该问题之前自己也不妨来温习一下 Python 中使用线程池的基本模式，下面的模板代码曾经是我的最爱：

```
1  import time
2  from concurrent.futures import ThreadPoolExecutor
3
4  def perform(x):
5      time.sleep(2)
6      print(f'process {x}')
7      return x + 1
8
9  with ThreadPoolExecutor(5) as executor:
10     for i in range(3):
11         executor.submit(perform, i)
12
13 print('done')
```

在上面的 `with` 上下文中会进行以下几步

1. 创建线程池
2. 提交任务到线程池
3. 等待所有任务完成
4. 最后关闭线程池，并执行后面的代码

所以执行后的输出大概如下：

```
process 2
process 0
process 1
done
```

前三行的输出顺序不定，它们由线程池中的线程执行的，done 一定是在所有任务完成了最后输出的。

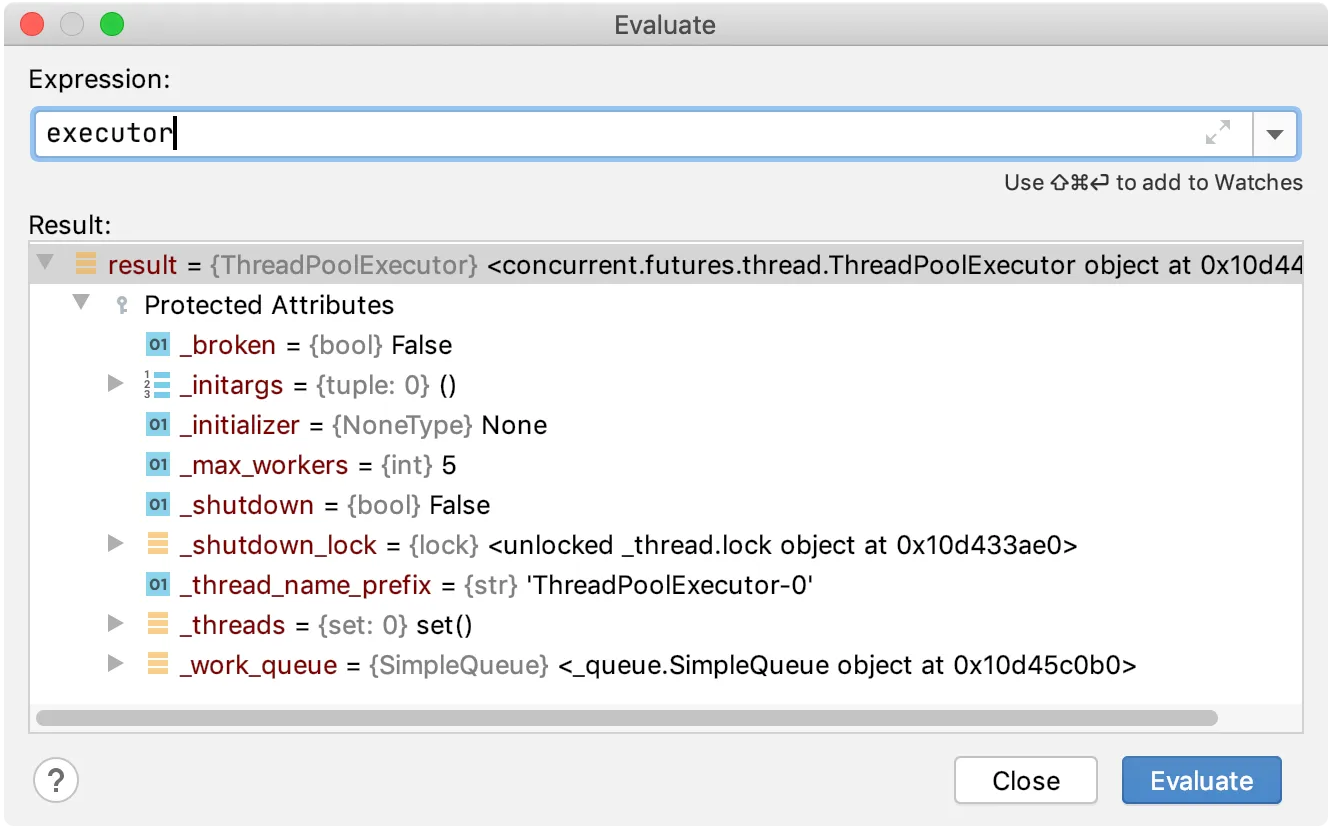
如果要收集子任务的输出，可以提交任务时放到 futures 列表中，如

```
futures.append(executor.submit(perform, i))
for futures in futures:
    print(future.result)
```

或是 `executor.map()` 一步提交多个任务并收集结果

```
results = executor.submit(perform, [0, 1, 2])
for result in results:
    print(result)
```

回顾完了我们重新回到正题上来，如果不停的向线程池提交任务，待执行的任务全部要积压在线程池的工作队列中，提交多了快了，远远超出了线程池的处理速度就会迅速把本地内存挤暴。来观察一下一个 ThreadPoolExecutor 的内部属性

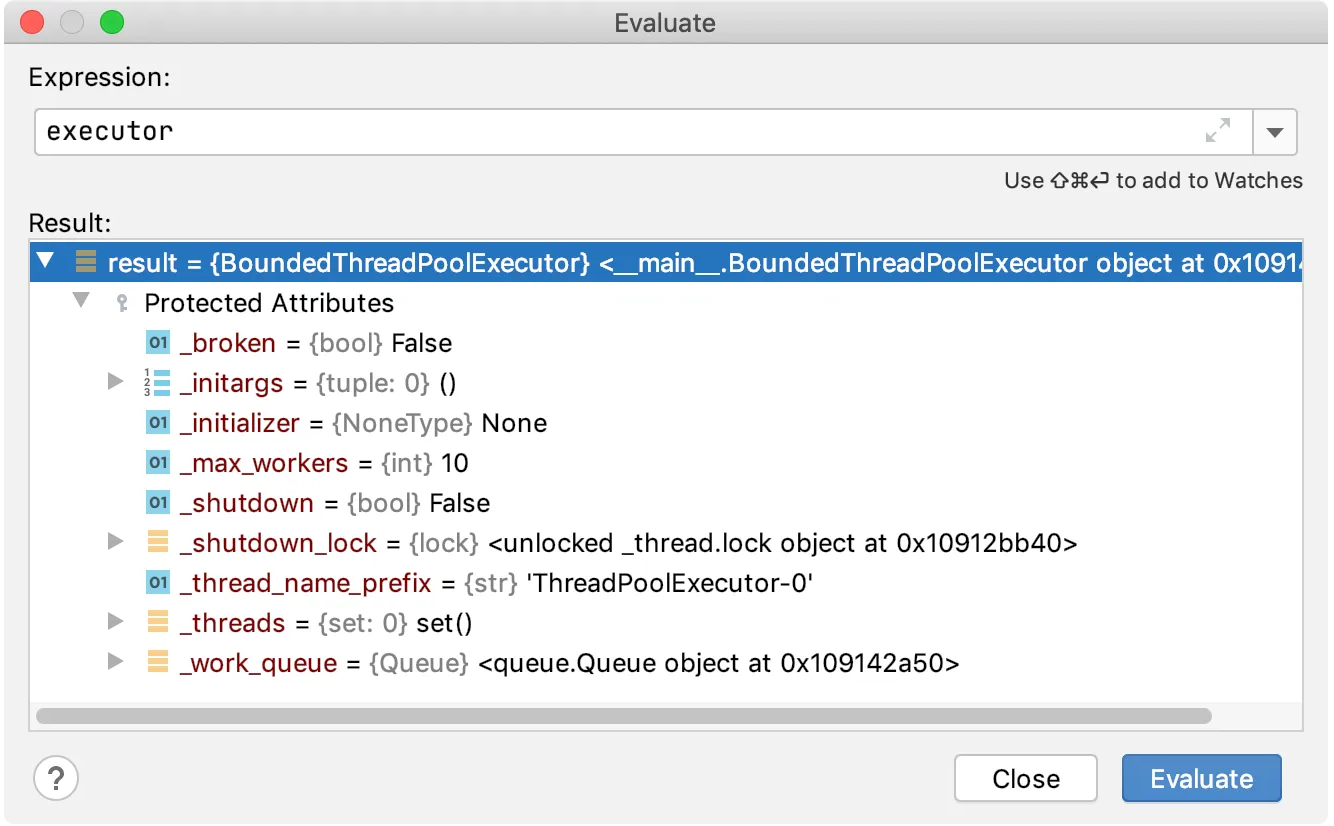


注意到它有一个 `_work_queue` 用来存放通过 `submit` 提交的待处理任务，默认实现为 `SimpleQueue`，而 `queue.SimpleQueue` 是一个没有节制的队列，你可以一直往里面添加记录，只要计数没溢出并且内存充足。这就是简单使用 ThreadPoolExecutor 造成 OutOfMemory 的元凶，更可恶的是 Python 不坦诚的告诉我们内存不足，而是直接把系统拖死。

因此解决办法就是要想法设法在提交任务之前检查当前 `_work_queue` 大小是否超过某个限定值，是的话等降下来后再提交新任务。然而 `_work_queue` 是一个私有变量，不允许从外部访问，那么索性我们创造一个子类把 `_work_queue` 置换成一个有容量限制的 Queue, 那就是 `queue.Queue`. `BoundedThreadPoolExecutor` 并使用方式如下：

```
import time
from concurrent.futures import ThreadPoolExecutor
from queue import Queue

def perform(x):
    time.sleep(2)
    print(f'process {x}')

class BoundedThreadPoolExecutor(ThreadPoolExecutor):
    def __init__(self, max_workers, max_waiting_tasks, *args, **kwargs):
        super().__init__(max_workers=max_workers, *args, **kwargs)
        self._work_queue = Queue(maxsize=max_waiting_tasks)

with BoundedThreadPoolExecutor(10, 100) as executor:
    for i in range(999999999):
        executor.submit(perform, i)

print('done')
```

现在操作系统再也不用担心过多的任务会把内存消耗干净了。再来看一下 `BoundedThreadPoolExecutor` 内部



`_work_queue` 变成了 `queue.Queue` 实现，原理是借助于 Queue 可指定 `maxsize`，当 Queue 的大小达到这个值，提交任务时将被阻塞，直致工作线程从中取走待执行任务，Queue 的大小低于 `maxsize` 值，才能继续往里头提交任务，这达到了一种流量控制的效果。

链接：

1. ThreadPoolExecutor: how to limit the queue maxsize?

类别: Python. 标签: multithread. 阅读(212). 订阅评论. TrackBack.