

人怜直节生来瘦，自许高材老更刚。

专注web、移动应用安全研究。

博客园

首页

新随笔

联系

订阅

管理

随笔 - 138 文章 - 8 评论 - 49

昵称：bamb00

园龄：5年9个月

粉丝：89

关注：1

+加关注

< 2016年12月 >						
日	一	二	三	四	五	六
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

搜索

找找看

谷歌搜索

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

我的标签

python(16)

uliweb(5)

django(3)

xss(3)

linux(3)

excel(2)

漏洞(2)

面试题(2)

url(2)

URL提取(1)

更多

线程池原理及python实现

为什么需要线程池

目前的大多数网络服务器，包括Web服务器、Email服务器以及数据库服务器等都具有一个共同点，就是单位时间内必须处理数目巨大的连接请求，但处理时间却相对较短。

传统多线程方案中我们采用的服务器模型则是一旦接受到请求之后，即创建一个新的线程，由该线程执行任务。任务执行完毕后，线程退出，这就是是“即时创建，即时销毁”的策略。尽管与创建进程相比，创建线程的时间已经大大的缩短，但是如果提交给线程的任务是执行时间较短，而且执行次数极其频繁，那么服务器将处于不停的创建线程，销毁线程的状态。

我们将传统方案中的线程执行过程分为三个过程：T1、T2、T3：

- T1：线程创建时间
- T2：线程执行时间，包括线程的同步等时间
- T3：线程销毁时间

那么我们可以看出，线程本身的开销所占的比例为(T1+T3) / (T1+T2+T3)。如果线程执行的时间很短的话，这比开销可能占到20%-50%左右。如果任务执行时间很频繁的话，这笔开销将是不可忽略的。

除此之外，线程池能够减少创建的线程个数。通常线程池所允许的并发线程是有上界的，如果同时需要并发的线程数超过上界，那么一部分线程将会等待。而传统方案中，如果同时请求数目为2000，那么最坏情况下，系统可能需要产生2000个线程。尽管这不是一个很大的数目，但是也有部分机器可能达不到这种要求。

因此线程池的出现正是着眼于减少线程池本身带来的开销。线程池采用预创建的技术，在应用程序启动之后，将立即创建一定数量的线程(N1)，放入空闲队列 中。这些线程都是处于阻塞（Suspended）状态，不消耗CPU，但占用较小的内存空间。当任务到来后，缓冲池选择一个空闲线程，把任务传入此线程中运行。当N1个线程都在处理任务后，缓冲池自动创建一定数量的新线程，用于处理更多的任务。在任务执行完毕后线程也不退出，而是继续保持在池中等待下一次的任务。当系统比较空闲时，大部分线程都一直处于暂停状态，线程池自动销毁一部分线程，回收系统资源。

基于这种预创建技术，线程池将线程创建和销毁本身所带来的开销分摊到了各个具体的任务上，执行次数越多，每个任务所分担到的线程本身开销则越小，不过我们另外可能要考虑进去线程之间同步所带来的开销。

构建线程池框架


一般线程池都必须具备下面几个组成部分：

- 线程池管理器:用于创建并管理线程池
- 工作线程: 线程池中实际执行的线程
- 任务接口: 尽管线程池大多数情况下是用来支持网络服务器，但是我们将线程执行的任务抽象出来，形成任务接口，从而是的线程池与具体的任务无关。
- 任务队列:线程池的概念具体到实现则可能是队列，链表之类的数据结构，其中保存执行线程。

我们把任务放进队列中去，然后开 N 个线程，每个线程都去队列中取一个任务，执行完了之后告诉系统说我执行完了，然后接着去队列中取下一个任务，直至队列中所有任务取空，退出线程。

这就是一般的线程池实现的原理，下面看一个实际的代码：

线程池的python实现代码：



```
1 # !/usr/bin/env python
2 # -*- coding:utf-8 -*-
3
4 import Queue
5 import threading
6 import time
7
```

随笔分类
android安全(44)
Android底层(5)
android开发(4)
C/C++(3)
django(4)
Linux(5)
python(40)
WEB安全(13)
数据结构与算法(4)
随笔档案
2016年6月 (1)
2016年4月 (1)
2016年3月 (1)
2016年2月 (1)
2016年1月 (2)
2015年10月 (1)
2015年9月 (2)
2015年8月 (2)
2015年7月 (1)
2015年4月 (3)
2015年1月 (3)
2014年11月 (2)
2014年10月 (6)
2014年9月 (8)
2014年8月 (3)
2014年7月 (4)
2014年3月 (1)
2014年2月 (2)
2014年1月 (1)
2013年12月 (1)
2013年11月 (5)

```
8 class WorkManager(object):
9     def __init__(self, work_num=1000,thread_num=2:
10         self.work_queue =Queue.Queue()
11         self.threads = []
12         self.__init_work_queue(work_num)
13         self.__init_thread_pool(thread_num)
14
15     """
16     初始化线程
17     """
18     def __init_thread_pool(self,thread_num):
19         for i in range(thread_num):
20             self.threads.append(Work(self.work_queue))
21
22     """
23     初始化工作队列
24     """
25     def __init_work_queue(self, jobs_num):
26         for i in range(jobs_num):
27             self.add_job(do_job, i)
28
29     """
30     添加一项工作入队
31     """
32     def add_job(self, func, *args):
33         self.work_queue.put((func, list(args)))#任务入队, Queue内部实现了同步机制
34
35     """
36     等待所有线程运行完毕
37     """
38     def wait_allcomplete(self):
39         for item in self.threads:
40             if item.isAlive():item.join()
41
42 class Work(threading.Thread):
43     def __init__(self, work_queue):
44         threading.Thread.__init__(self)
45         self.work_queue =work_queue
46         self.start()
47
48     def run(self):
49         #死循环, 从而让创建的线程在一定条件下关闭退出
50         while True:
51             try:
52                 do, args = self.work_queue.get(block=False)#任务异步出队, Queue内部实现了
53                 do(args)
54                 self.work_queue.task_done(#通知系统任务完成
55             except:
56                 break
57
58 #具体要做的任务
59 def do_job(args):
60     time.sleep(0.1)#模拟处理时间
61     print threading.current_thread(), list(args)
62
63 if __name__ == '__main__':
64     start = time.time()
65     work_manager = WorkManager(10000, 10#或者work_manager = WorkManager(10000, 20)
66     work_manager.wait_allcomplete()
67     end = time.time()
68     print "cost all time: %s" % (end-start)
```

Work类是一个Python线程池,不断地从workQueue队列中获取需要执行的任务,执行之,并将结果写入到resultQueue中。这里的workQueue和resultQueue都是线程安全的,其内部对各个线程的操作做了互斥。当从workQueue中获取任务超时,则线程结束。

WorkerManager负责初始化Python线程池,提供将任务加入队列和获取结果的接口,并能等待所有任务完成。

在 Python 中使用线程时,这个模式是一种很常见的并且推荐使用的方式。具体工作步骤描述如下:

- 1. 创建一个 Queue.Queue() 的实例,然后使用数据对它进行填充。
- 2. 将经过填充数据的实例传递给线程类,后者是通过继承 threading.Thread 的方式创建的。

2013年10月 (15)
2013年9月 (5)
2013年8月 (12)
2013年7月 (18)
2013年6月 (8)
2013年5月 (5)
2013年4月 (7)
2013年3月 (1)
2011年12月 (2)
2011年10月 (1)
2011年8月 (1)
2011年7月 (1)
2011年6月 (3)
2011年5月 (1)
2011年4月 (4)
2011年3月 (3)

友情链接

Only3nd's blog
9Hao's blog
APTTECH
c32's blog
sh4d0w
sunysen
暗影's Blog
知道创宇Fooying的博客

最新评论

1. Re:python面试题大全（一）
抱走。谢谢楼主
--林深时见鹿
2. Re:线程池原理及python实现
写的很好，慢慢看:)
--duohappy
3. Re:Android studio动态调试smali
您好，这个方法我试了很多遍。但我自己始终是出现了了如图所示的问题，不知道您知道问题出在么？
--TellMeUp

- 3. 生成守护线程池。
- 4. 每次从队列中取出一个项目，并使用该线程中的数据和 run 方法以执行相应的工作。
- 5. 在完成这项工作之后，使用 queue.task_done() 函数向任务已经完成的队列发送一个信号。
- 6. 对队列执行 join 操作，实际上意味着等到队列为空，再退出主程序。

在使用这个模式时需要注意一点：通过将守护线程设置为 true，将允许主线程或者程序仅在守护线程处于活动状态时才能够退出。这种方式创建了一种简单的方式以控制程序流程，因为在退出之前，您可以对队列执行 join 操作、或者等到队列为空。队列模块文档详细说明了实际的处理过程，请参见参考资料：

join()


保持阻塞状态，直到处理了队列中的所有项目为止。在将一个项目添加到该队列时，未完成的任务的总数就会增加。当使用者线程调用 task_done() 以表示检索了该项目、并完成了所有的工作时，那么未完成的任务的总数就会减少。当未完成的任务的总数减少到零时，join() 就会结束阻塞状态。

参考：<http://blog.csdn.net/yatere/article/details/7316487>
<http://blog.csdn.net/liu1pan2min3/article/details/8545979>
<http://www.ibm.com/developerworks/cn/aix/library/au-threadingpython/?ca=drs-tp3008>

分类: [python](#)

好文要顶 关注我 收藏该文



 bamb00

关注 - 1

粉丝 - 89

[+加关注](#)

1 0

« 上一篇: [python基础——类定义（转）](#)
» 下一篇: [python中的*args和**kwargs](#)

posted @ 2013-10-09 19:42 bamb00 阅读(4365) 评论(1) 编辑 收藏

评论列表

#1楼 2016-10-27 21:11 duohappy

写的很好，慢慢看:)

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

最新IT新闻：

- 星巴克推出原创系列动画，这和多卖一杯咖啡有什么关系？
 - 苹果设柏林分支机构 专门负责挖Here地图人才
 - 20年前的科技头条：人们最爱56K拨号上网，网络电视首次亮相
 - 不管易到用车怎么否认欠款 但是它们的客服电话真的停掉了
 - 明年Android生态将走向何方？这七大趋势给你答案
- » 更多新闻...

最新知识库文章：

- 写给未来的程序媛
 - 高质量的工程代码为什么难写
 - 循序渐进地代码重构
 - 技术的正宗与野路子
 - 陈皓：什么是工程师文化？
- » 更多知识库文章...

4. Re:破解android手机图形锁

能加微信或者qq多交流吗 我也是android安全研究员，在数字公司，1043576348 这个连上手机你首先要在手机上确认才行啊。

--Newzq

5. Re:*管家病毒查杀模块逆向分析

@calvinK呵呵...

--bamb00

阅读排行榜

1. C++类型转换总结(30821)

2. python面试题大全（一）(8138)

3. c++ 函数返回引用(7436)

4. ZjDroid工具介绍及脱壳详细示例(5905)

5. 线程池原理及python实现(4365)

评论排行榜

1. 对一个伪装成微信的加固病毒的分析(8)

2. 反编译apk插入广告（一）(4)

3. boot.img的修改(4)

4. “逃离大厦”游戏的破解(4)

5. python面试题大全（一）(3)

推荐排行榜

1. 对一个伪装成微信的加固病毒的分析(11)

2. C++类型转换总结(10)

3. boot.img的修改(5)

4. *管家病毒查杀模块逆向分析(5)

5. Apk去签名校验详解(5)