二

14 为什么不应该自动创建线程池?

在本课时我们主要学习为什么不应该自动创建线程池，所谓的自动创建线程池就是直接调用 Executors 的各种方法来生成前面学过的常见的线程池，例如 Executors.newCachedThreadPool()。但这样做是有一定风险的，接下来我们就来逐一分析自动创建线程池可能带来哪些问题。

FixedThreadPool

首先我们来看第一种线程池 FixedThreadPool， 它是线程数量固定的线程池，如源码所示，newFixedThreadPool 内部实际还是调用了 ThreadPoolExecutor 构造函数。

```
public static ExecutorService newFixedThreadPool(int nThreads) {

    return new ThreadPoolExecutor(nThreads, nThreads,0L, TimeUnit.MILLISECONDS,new

}
```

通过往构造函数中传参，创建了一个核心线程数和最大线程数相等的线程池，它们的数量也就是我们传入的参数，这里的重点是使用的队列是容量没有上限的 LinkedBlockingQueue，如果我们对任务的处理速度比较慢，那么随着请求的增多，队列中堆积的任务也会越来越多，最终大量堆积的任务会占用大量内存，并发生 OOM ， 也就是 OutOfMemoryError，这几乎会影响到整个程序，会造成很严重的后果。

SingleThreadExecutor

第二种线程池是 SingleThreadExecutor，我们来分析下创建它的源码。

```
public static ExecutorService newSingleThreadExecutor() {

    return new FinalizableDelegatedExecutorService (new ThreadPoolExecutor(1, 1,0L,

}
```

你可以看出，newSingleThreadExecutor 和 newFixedThreadPool 的原理是一样的，只不

过把核心线程数和最大线程数都直接设置成了 1，但是任务队列仍是无界的 LinkedBlockingQueue，所以也会导致同样的问题，也就是当任务堆积时，可能会占用大量的内存并导致 OOM。

CachedThreadPool

第三种线程池是 CachedThreadPool，创建它的源码下所示。

```
public static ExecutorService newCachedThreadPool() {
    return new ThreadPoolExecutor(0, Integer.MAX_VALUE,60L, TimeUnit.SECONDS,new Sy
}
```

这里的 CachedThreadPool 和前面两种线程池不一样的地方在于任务队列使用的是 SynchronousQueue，SynchronousQueue 本身并不存储任务，而是对任务直接进行转发，这本身是没有问题的，但你会发现构造函数的第二个参数被设置成了 Integer.MAX_VALUE，这个参数的含义是最大线程数，所以由于 CachedThreadPool 并不限制线程的数量，当任务数量特别多的时候，就可能会导致创建非常多的线程，最终超过了操作系统的上限而无法创建新线程，或者导致内存不足。

ScheduledThreadPool 和 SingleThreadScheduledExecutor

第四种线程池 ScheduledThreadPool 和第五种线程池 SingleThreadScheduledExecutor 的原理是一样的，创建 ScheduledThreadPool 的源码如下所示。

```
public static ScheduledExecutorService newScheduledThreadPool(int corePoolSize) {
    return new ScheduledThreadPoolExecutor(corePoolSize);
}
```

而这里的 ScheduledThreadPoolExecutor 是 ThreadPoolExecutor 的子类，调用的它的构造方法如下所示。

```
public ScheduledThreadPoolExecutor(int corePoolSize) {
    super(corePoolSize, Integer.MAX_VALUE, 0, NANOSECONDS,new DelayedWorkQueue());
}
```

我们通过源码可以看出，它采用的任务队列是 DelayedWorkQueue，这是一个延迟队列，

同时也是一个无界队列，所以和 LinkedBlockingQueue 一样，如果队列中存放过多的任务，就可能导致 OOM。

你可以看到，这几种自动创建的线程池都存在风险，相比较而言，我们自己手动创建会更好，因为我们可以更加明确线程池的运行规则，不仅可以选择适合自己的线程数量，更可以在必要的时候拒绝新任务的提交，避免资源耗尽的风险。

上一页 下一页