二

15 合适的线程数量是多少？CPU 核心数和线程数的关系?

在本课时我们主要学习合适的线程数量是多少，以及 CPU 核心数和线程数的关系。

你可能经常在面试中被问到这两个问题，如果想要很好地回答它们首先你需要了解，我们调整线程池中的线程数量的最主要的目的是为了充分并合理地使用 CPU 和内存等资源，从而最大限度地提高程序的性能。在实际工作中，我们需要根据任务类型的不同选择对应的策略。

CPU 密集型任务

首先，我们来看 CPU 密集型任务，比如加密、解密、压缩、计算等一系列需要大量耗费 CPU 资源的任务。对于这样的任务最佳的线程数为 CPU 核心数的 1~2 倍，如果设置过多的线程数，实际上并不会起到很好的效果。此时假设我们设置的线程数量是 CPU 核心数的 2 倍以上，因为计算任务非常重，会占用大量的 CPU 资源，所以这时 CPU 的每个核心工作基本都是满负荷的，而我们又设置了过多的线程，每个线程都想去利用 CPU 资源来执行自己的任务，这就会造成不必要的上下文切换，此时线程数的增多并没有让性能提升，反而由于线程数量过多会导致性能下降。

针对这种情况，我们最好还要同时考虑在同一台机器上还有哪些其他会占用过多 CPU 资源的程序在运行，然后对资源使用做整体的平衡。

耗时 IO 型任务

第二种任务是耗时 IO 型，比如数据库、文件的读写，网络通信等任务，这种任务的特点是并不会特别消耗 CPU 资源，但是 IO 操作很耗时，总体会占用比较多的时间。对于这种任务最大线程数一般会大于 CPU 核心数很多倍，因为 IO 读写速度相比于 CPU 的速度而言是比较慢的，如果我们设置过少的线程数，就可能导致 CPU 资源的浪费。而如果我们设置更多的线程数，那么当一部分线程正在等待 IO 的时候，它们此时并不需要 CPU 来计算，那么另外的线程便可以利用 CPU 去执行其他的任务，互不影响，这样的话在任务队列中等待的任务就会减少，可以更好地利用资源。

《Java并发编程实战》的作者 Brain Goetz 推荐的计算方法：

```
线程数 = CPU 核心数 *（1+平均等待时间/平均工作时间）
```

通过这个公式，我们可以计算出一个合理的线程数量，如果任务的平均等待时间长，线程数就随之增加，而如果平均工作时间长，也就是对于我们上面的 CPU 密集型任务，线程数就随之减少。

太少的线程数会使得程序整体性能降低，而过多的线程也会消耗内存等其他资源，所以如果想要更准确的话，可以进行压测，监控 JVM 的线程情况以及 CPU 的负载情况，根据实际情况衡量应该创建的线程数，合理并充分利用资源。

结论

综上所述我们就可以得出一个结论：

- 线程的平均工作时间所占比例越高，就需要越少的线程；
- 线程的平均等待时间所占比例越高，就需要越多的线程；
- 针对不同的程序，进行对应的实际测试就可以得到最合适的选择。

上一页 下一页