二

08 为什么多线程会带来性能问题?

在本课时我们主要学习为什么多线程会带来性能问题?

什么是性能问题

在上一课时我们已经学习了多线程带来的线程安全问题，但对于多线程而言，它不仅可能会带来线程安全问题，还有可能会带来性能问题，也许你会奇怪，我们使用多线程的最大目的不就是为了提高性能吗？让多个线程同时工作，加快程序运行速度，为什么反而会带来性能问题呢？这是因为单线程程序是独立工作的，不需要与其他线程进行交互，但多线程之间则需要调度以及合作，调度与合作就会带来性能开销从而产生性能问题。

首先，我们来了解究竟什么是性能问题？其实性能问题有许多的表现形式，比如服务器的响应慢、吞吐量低、内存占用过多就属于性能问题。我们设计优秀的系统架构、购置更多的 CDN 服务器、购买更大的带宽等都是为了提高性能，提高用户体验，虽然运行速度慢不会带来严重的后果，通常只需要我们多等几秒就可以，但这会严重影响用户的体验。有研究表明，页面每多响应 1 秒，就会流失至少 7% 的用户，而超过 8 秒无法返回结果的话，几乎所有用户都不会选择继续等待。我们引入多线程的一大重要原因就是想提高程序性能，所以不能本末倒置，不能因为引入了多线程反而程序运行得更慢了，所以我们必须要解决多线程带来的性能问题。

为什么多线程会带来性能问题

那么什么情况下多线程编程会带来性能问题呢？主要有两个方面，一方面是线程调度，另一个方面是线程协作。

调度开销

上下文切换

首先，我们看一下线程调度，在实际开发中，线程数往往是大于 CPU 核心数的，比如

CPU 核心数可能是 8 核、16 核，等等，但线程数可能达到成百上千个。这种情况下，操作系统就会按照一定的调度算法，给每个线程分配时间片，让每个线程都有机会得到运行。而在进行调度时就会引起上下文切换，上下文切换会挂起当前正在执行的线程并保存当前的状态，然后寻找下一处即将恢复执行的代码，唤醒下一个线程，以此类推，反复执行。但上下文切换带来的开销是比较大的，假设我们的任务内容非常短，比如只进行简单的计算，那么就有可能发生我们上下文切换带来的性能开销比执行线程本身内容带来的开销还要大的情况。

缓存失效

不仅上下文切换会带来性能问题，缓存失效也有可能带来性能问题。由于程序有很大概率会再次访问刚才访问过的数据，所以为了加速整个程序的运行，会使用缓存，这样我们在使用相同数据时就可以很快地获取数据。可一旦进行了线程调度，切换到其他线程，CPU就会去执行不同的代码，原有的缓存就很可能失效了，需要重新缓存新的数据，这也会造成一定的开销，所以线程调度器为了避免频繁地发生上下文切换，通常会给被调度到的线程设置最小的执行时间，也就是只有执行完这段时间之后，才可能进行下一次的调度，由此减少上下文切换的次数。

那么什么情况会导致密集的上下文切换呢？如果程序频繁地竞争锁，或者由于 IO 读写等原因导致频繁阻塞，那么这个程序就可能需要更多的上下文切换，这也就导致了更大的开销，我们应该尽量避免这种情况的发生。

协作开销

除了线程调度之外，线程协作同样也有可能带来性能问题。因为线程之间如果有共享数据，为了避免数据错乱，为了保证线程安全，就有可能禁止编译器和 CPU 对其进行重排序等优化，也可能出于同步的目的，反复把线程工作内存的数据 flush 到主存中，然后再从主内存 refresh 到其他线程的工作内存中，等等。这些问题在单线程中并不存在，但在多线程中为了确保数据的正确性，就不得不采取上述方法，因为线程安全的优先级要比性能优先级更高，这也间接降低了我们的性能。

上一页 下一页