## 一、多线程的机遇与挑战

### 1.1 多线程的挑战

#### 1.1.1 上下文切换

CPU通过时间片分配算法来循环执行任务，每个时间片非常短，通常是几十毫秒。当前任务执行一个时间片后会切换到下一个任务，在切换前会保存该任务的状态，以便下次切回这个任务时，可以再加载这个任务的状态。

任务从保存到再加载的过程就是一次上下文切换。

上下文切换会影响多线程的执行速度。

当每个任务执行的时间比较短的时候，多线程是没优势的，因为上下文切换的性能损耗影响比较大。

多线程竞争锁时，会进行上下文切换，有时可以通过无锁手段，如：无锁并发、CAS算法等，来减少上下文切换的次数。

#### 1.1.2 死锁

锁可以对临界区数据进行保护，确保数据的一致性。但锁的使用不当，会造成死锁，导致程序不可用。

避免死锁的常见方法有：避免一个线程同时获得多个锁；尽量保证一个锁只占用一个资源；使用定时锁，tryLock();保证锁出现的顺序一致。

#### 1.1.3 资源限制

程序的执行速度受限于计算机硬件资源(宽带上下载速度、硬盘读写速度、CPU处理速度)和软件资源(数据库连接数、socket连接数等)。

对于硬件资源限制，可以使用集群方式进行解决(同时会引入其他问题，如：分布式session、分布式事务等)；

对于软件资源限制，可以通过建立连接缓存池并且调整程序的并发度。

### 1.2 多线程的机遇

尽管存在着诸多挑战，但由于硬件技术的提升(多核CPU)和程序设计的复杂度提高，多线程的使用可以提高对硬件资源的利用率已经以更好的方式实现日益复杂的软件需求。

## 二、Java并发编程基础