# 多线程

## 线程执行

如果有两个任务需要处理，一个任务A，一个任务B

方案一：一个线程执行任务A和B，A执行完后，执行B

方案二：两个线程A和B去执行任务A 和 B，同时进行

哪个方案更快？

线程的执行，是由CPU进行调度的，一个CPU在同一时刻只会执行一个线程，我们看上去的线程A 和 线程B并发执行。

为了让用户感觉这些任务正在同时进行，操作系统利用了时间片轮转的方式，CPU给每个任务都服务一定的时间，然后把当前任务的状态保存下来，在加载下一任务的状态后，继续服务下一任务。任务的状态保存及再加载，这段过程就叫做上下文切换。

上下文切换过程是需要时间的；现在我们来看一下上面的问题，小伙伴们再看一下是哪个方案快呢？是不是有人会说方案一，因为不需要线程切换；方案二需要来回切换这两个线程，耗时会多点。

## 为什么要多线程

在我们真实业务中，我们是什么流程？

上图的流程：

1、先发起网络请求

2、Web服务器解析请求

3、请求后端的数据库获取数据

4、获取数据后，进行处理

5、把处理结果放回给用户

这个是我们处理业务的时候，常规的请求流程；我们看一下整个过程涉及到什么计算机处理。

1、网络请求----->网络IO

2、解析请求----->CPU

3、请求数据库----->网络IO

4、MySQL查询数据----->磁盘IO

5、MySQL返回数据----->网络IO

6、数据处理----->CPU

7、返回数据给用户----->网络IO

在真实业务中我们不单单会涉及CPU计算，还有网络IO和磁盘IO处理，这些处理是非常耗时的。如果一个线程整个流程是上图的流程，真正涉及到CPU的只有2个节点，其他的节点都是IO处理，那么线程在做IO处理的时候，CPU就空闲出来了，CPU的利用率就不高。

现在知道多线程的用处：就是为了提升CPU利用率。

## 提升QPS/TPS

衡量系统性能如何，主要指标系统的（QPS/TPS）

QPS/TPS：每秒能够处理请求/事务的数量

并发数：系统同时处理的请求/事务的数量

响应时间：就是平均处理一个请求/事务需要时长

QPS/TPS = 并发数/响应时间

上面公式代表并发数越大，QPS就越大；所以很多人就会以为调大线程池，并发数就会大，也会提升QPS，所以才会出现一开始前言所说的，大多数人的误区。

其实QPS还跟响应时间成反比，响应时间越大，QPS就会越小。

虽然并发数调大了，就会提升QPS，但线程数也会影响响应时间，因为上面我们也提到了上下文切换的问题，那怎么设置线程数的呢？

## 如何设置线程数

那我们如何分配线程？我们提供一个公式：

最佳线程数目 = （（线程等待时间+线程CPU时间）/线程CPU时间 ）\* CPU数目

备注这个公式也是前辈们分享的，当然之前看了淘宝前台系统优化实践的文章，和上面的公式很类似，不过在CPU数目那边，他们更细化了，上面的公式只是参考。不过不管什么公式，最终还是在生产环境中运行后，再优化调整。

我们继续上面的任务，我们的服务器CPU核数为4核，一个任务线程cpu耗时为20ms，线程等待（网络IO、磁盘IO）耗时80ms，那最佳线程数目：( 80 + 20 )/20 \* 4 = 20。也就是设置20个线程数最佳。

从这个公式上面我们就得出，线程的等待时间越大，线程数就要设置越大，这个正好符合我们上面的分析，可提升CPU利用率。那从另一个角度上面说，线程数设置多大，是根据我们自身的业务的，需要自己去压力测试，设置一个合理的数值。

## 基础常规标准

因为很多业务集中到一个线程池中，不像上面的案例比较简单，事实上业务太多，怎么设置呢？这个就是要去压力测试去调整。不过我们的前辈已经帮我们总结了一个基础的值（最终还是要看运行情况自行调整）

1、CPU密集型：操作内存处理的业务，一般线程数设置为：CPU核数 + 1 或者 CPU核数\*2。核数为4的话，一般设置 5 或 8

2、IO密集型：文件操作，网络操作，数据库操作，一般线程设置为：cpu核数 / (1-0.9)，核数为4的话，一般设置 40

# 线程通信

# 线程同步