# 线程池

## 什么是多线程？

线程是操作系统能够进行运算调度的最小单位，它被包含在进程之中，是进程中的实际运作单位。程序员可以通过它进行多处理器编程，你可以使用多线程对运算密集型任务提速。比如，如果一个线程完成一个任务要100毫秒，那么用十个线程完成改任务只需10毫秒。

一行代码执行流，完成一组代码的执行。

## 2、使用多线程的目的？

多线程的作用不仅仅是提升性能。实际上，多线程并不一定能提升性能（甚至还会降低性能）；多线程也不只是为了提升性能。多线程主要有以下的应用场景：

### 1、避免阻塞（异步调用）:

单个线程中的程序，是顺序执行的。如果前面的操作发生了阻塞，那么就会影响到后面的操作。这时候可以采用多线程，我感觉就等于是异步调用。这样的例子有很多：

Ajax调用，就是浏览器会启一个新的线程，不阻塞当前页面的正常操作； 流程在某个环节调用web service，如果是同步调用，则需要等待web service调用结果，可以启动新线程来调用，不影响主流程； 创建工单时，需要级联往其他表中插入数据，可以将级联插入的动作放到新线程中，先返回工单创建的结果……

### 2、避免CPU空转 :

以http server为例，如果只用单线程响应HTTP请求，即处理完一条请求，再处理下一条请求的话，CPU会存在大量的闲置时间 。

处理一条请求，经常涉及到RPC(通讯协议)、数据库访问、磁盘IO(读写)等操作，这些操作的速度比CPU慢很多，而在等待这些响应的时候，CPU却不能去处理新的请求，因此http server的性能就很差 。所以很多web容器，都采用对每个请求创建新线程来响应的方式实现，这样在等待请求A的IO操作的等待时间里，就可以去继续处理请求B，对并发的响应性就好了很多 。

Cpu的工作：



### 3、提升性能

在满足条件的前提下，多线程确实能提升性能

打一个比方，多线程就相当于，把要炒的菜放到了不同的锅里，然后用不同的炉来炒，当然速度会比较快。本来需要先炒西红柿，10分钟；再炒白菜10分钟；加起来就需要20分钟。用了多线程以后，分别放在2个锅里炒，10分钟就都炒好了。

## 3、使用多线程需要满足3个条件：

1、任务具有并发性，也就是可以拆分成多个子任务。并不是什么任务都能拆分的，条件还比较苛刻 。子任务之间不能有先后顺序的依赖，必须是允许并行的 。

a=b+c,d=e+f; √ a=b+c,d=a+f;×

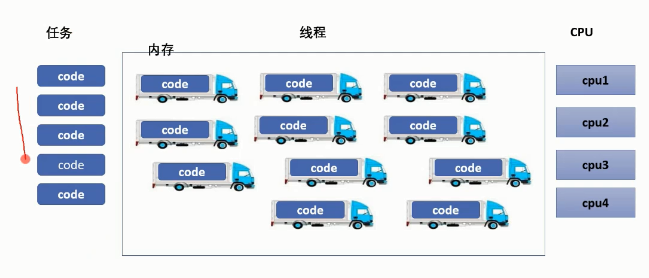
另外，还不能有资源竞争。比如2个线程都需要写一个文件，第1个线程将文件锁定了，第2个线程只能等着，这样的2个子任务，也不具备并发性；执行sychronized代码，也是同样的情况 。

2、只有在CPU是性能瓶颈的情况下，多线程才能实现提升性能的目的。比如一段程序，瓶颈在于IO操作，那么把这个程序拆分到2个线程中执行，也是无法提升性能的 。

3、需要有多核CPU才行。否则的话，虽然拆分成了多个可并行的子任务，但是没有足够的CPU，还是只有一个CPU在多个线程中切换来切换去，不但达不到提升性能的效果，反而由于增加了额外的开销，而降低了性能。类似于虽然把菜放到了2个锅里，但是只有1个炉子一样 。

## 4、大量并发的程序设计，开多少线程比较合适

### 1、思考：



结合上图，代码装到线程中，cpu用来执行

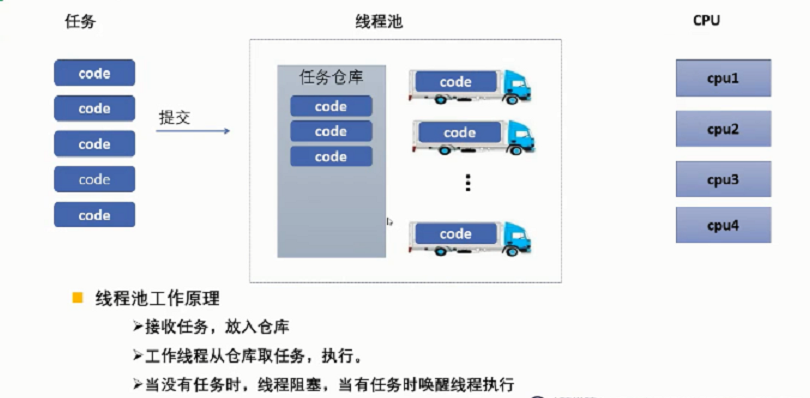
### 2、线程是不是越多越好？

线程在java中是一个对象，每一个java线程都需要一个操作系统线程的支持，线程的创建，销毁，都需要时间。 如果创建时间+销毁时间>执行任务的时间。就很不合算了

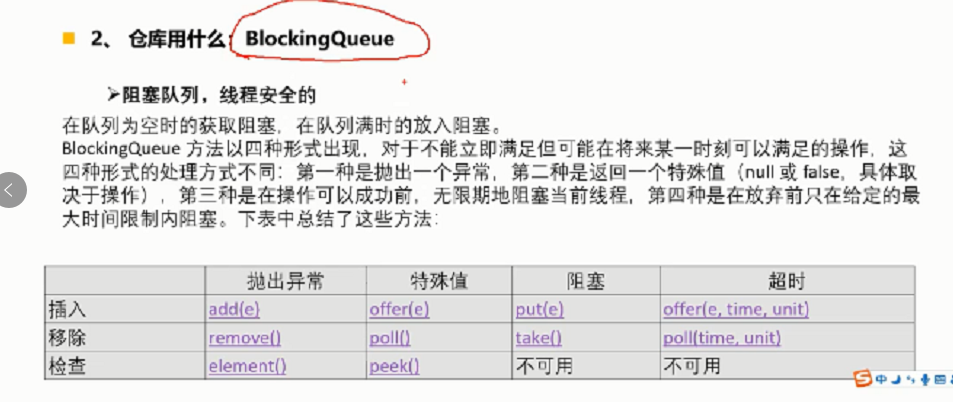
Java对象占用堆内存，操作系统线程占用系统内存，根据jvm规范，一个线程默认最大栈大小为1M，这个栈空间是需要从系统内存中分配的。线程过多，会损耗很多内存。

## 线程池原理

### 1、原理：



### 队列（仓库）的选择：



### 确定合适数量的线程

根据服务器的性能来找到最合适的数量

## 如何使用线程池

### 1、方法分析



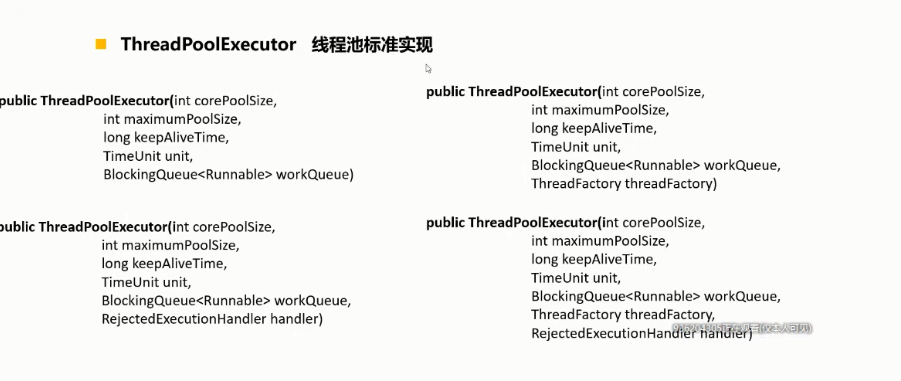
### 2、常用分析



3、定时任务的支持



### 3、线程池的标准实现



方法一：定义核心线程数、最大线程数、线程存活时间（只对非核心线程起作用）、时间单位、以及队列（仓库）、创建线程池的工厂

最大线程数是包含了核心线程数的；非核心线程数是不会被创建的，在线程池中一定会在队列的数量满了之后才会创建非核心线程

