thread_pool 、 process_pool sharing

什么是池

池的描述和定义：Pool（池）的概念被广泛的应用在服务器端软件的开发上。使用池结构可以明显的提高你的应用程序的速度，改善效率和降低系统资源的开销。所以在应用服务器端的开发中池的设计和实现是开发工作中的重要一环。

那么到底什么是池呢？我们可以简单的想象一下应用运行时的环境，当大量的客户并发的访问应用服务器时我们如何提供服务呢？我们可以为每一个客户提供一个新的服务对象进行服务。这种方法看起来简单，在实际应用中如果采用这种实现会有很多问题，**显而易见的是不断的创建和销毁新服务对象必将给造成系统资源的巨大开销**，导致系统的性能下降。

针对这个问题我们采用池的方式。池可以想象成就是一个容器保存着各种我们需要的对象。我们对这些对象进行复用，从而提高系统性能。从结构上看，它应该具有容器对象和具体的元素对象。从使用方法上看，我们可以直接取得池中的元素来用，也可以把我们要做的任务交给它处理。

一、什么是线程池

> 业务逻辑与网络处理相分离
> 作用一：不需要每次创建线程，减少线程创建销毁，
> 作用二：
> 异步解耦的作用，logInfo(”xxxxx”) 将task放到一个队列中，再起线程池去队列中拿任务写磁盘

1、所谓线程池

线程池其实是一种池化的技术的实现，池化技术的核心思想其实就是实现资源的一个复用，避免资源的重复创建和销毁带来的性能开销。在线程池中，线程池可以管理一堆线程，让线程执行完任务之后不会进行销毁，而是继续去处理其它线程已经提交的任务。

一般情况下，我们需要异步执行一些任务，这些任务的产生和执行是存在于我们程序的整个生命周期的，基本要求是当有任务需要执行时，这些线程可以自动拿到任务去执行，没有任务时这些线程处于阻塞或者睡眠状态。

1.1、线程池的好处:

- 降低资源消耗。通过重复利用已创建的线程降低线程创建和销毁造成的消耗。
- 提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。
- 提高线程的可管理性。线程是稀缺资源，如果无限制的创建，不仅会消耗系统资源，还会降低系统 的稳定性，使用线程池可以进行统一的分配，调优和监控。

1.2 线程池的工作原理

有子线程都运行着相同的代码。当有新的任务到来时，主线程将通过某种方式选择线程池中的某一个子线程来为之服务。相比与动态的创建子线程，选择一个已经存在的子线程的代价显然要小得多。至于主线程选择哪个子线程来为新任务服务，则有多种方式：

A.主线程使用某种算法来主动选择子线程。最简单、最常用的算法是随机算法和 Round Robin（轮流
选取）算法，但更优秀、更智能的算法将使任务在各个工作线程中更均匀地分配，从而减轻服务器
的整体压力。

B.主线程和所有子线程通过一个共享的工作队列来同步，子线程都睡眠在该工作队列上。当有新的任
务到来时，主线程将任务添加到工作队列中。这将唤醒正在等待任务的子线程，不过只有一个子线
程将获得新任务的”接管权“，它可以从工作队列中取出任务并执行之，而其他子线程将继续睡眠在
工作队列上。

2、线程池模型

> 主线程(放入到工作队列中)+工作队列+线程池(任务线程)+完成队列+主线程(从完成队列取)

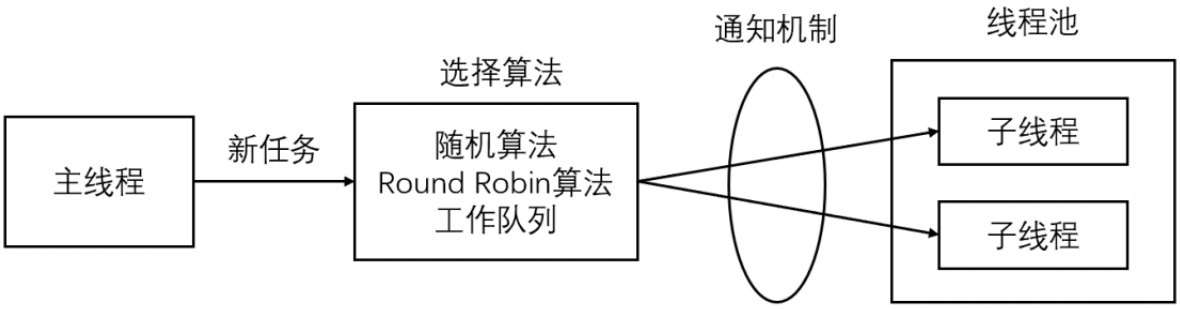
既然在程序生命周期内会产生很多任务，那么这些任务必须有一个存放的地方，而这个地方就是**队列**，所以不要一提到队列就认为是一个具体的 list，它可以是一个全局变量、链表等等。而线程池中的线程从队列中如何取任务，则也可以设计得非常灵活，如从尾部放入任务，从头部取出，或者从头部放入，从尾部取出等等。而队列也可以根据实际应用设计得”丰富多彩“，如可以根据任务得优先级，设计多个队列（如分为高中低三个级别、分为关键和普通两个级别）。

这本质上就是生产者消费者模式，产生任务的线程是生产者，线程池中的线程是消费者。当然，这不是绝对的，线程池中的线程处理一个任务以后可能会产生一个新的关联任务，那么此时这个工作线程又是生产者的角色。

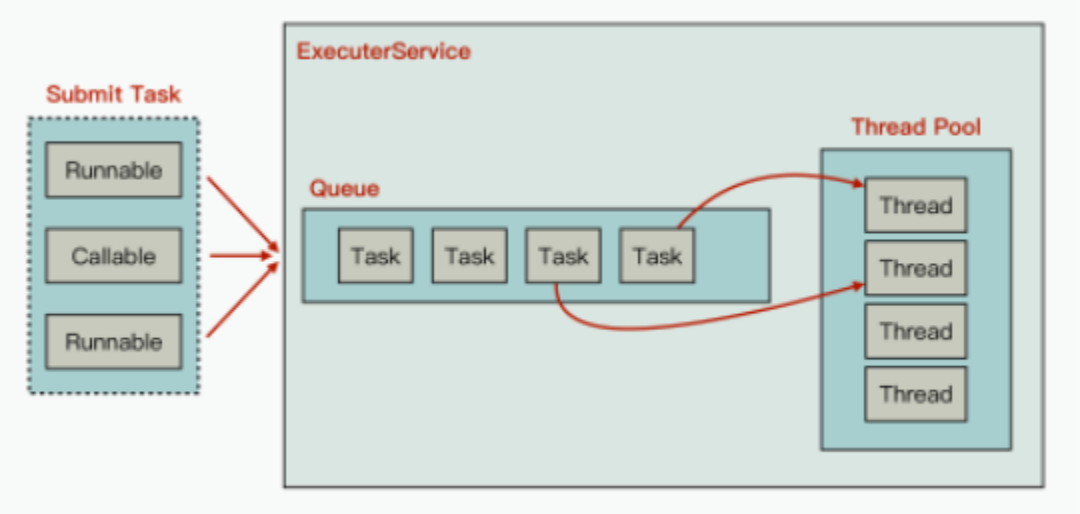
既然会有多个线程同时操作这个队列，根据多线程程序的原则，这个队列我们一般需要对其加锁，以避免多线程竞争产生非预期的结果。

当然，技术上除了要解决线程池的创建、往队列中投递任务、从队列中取任务处理，我们还需要做一些善后工作，如线程池的清理，即如何退出线程池中的工作线程和清理任务队列。

模型A.



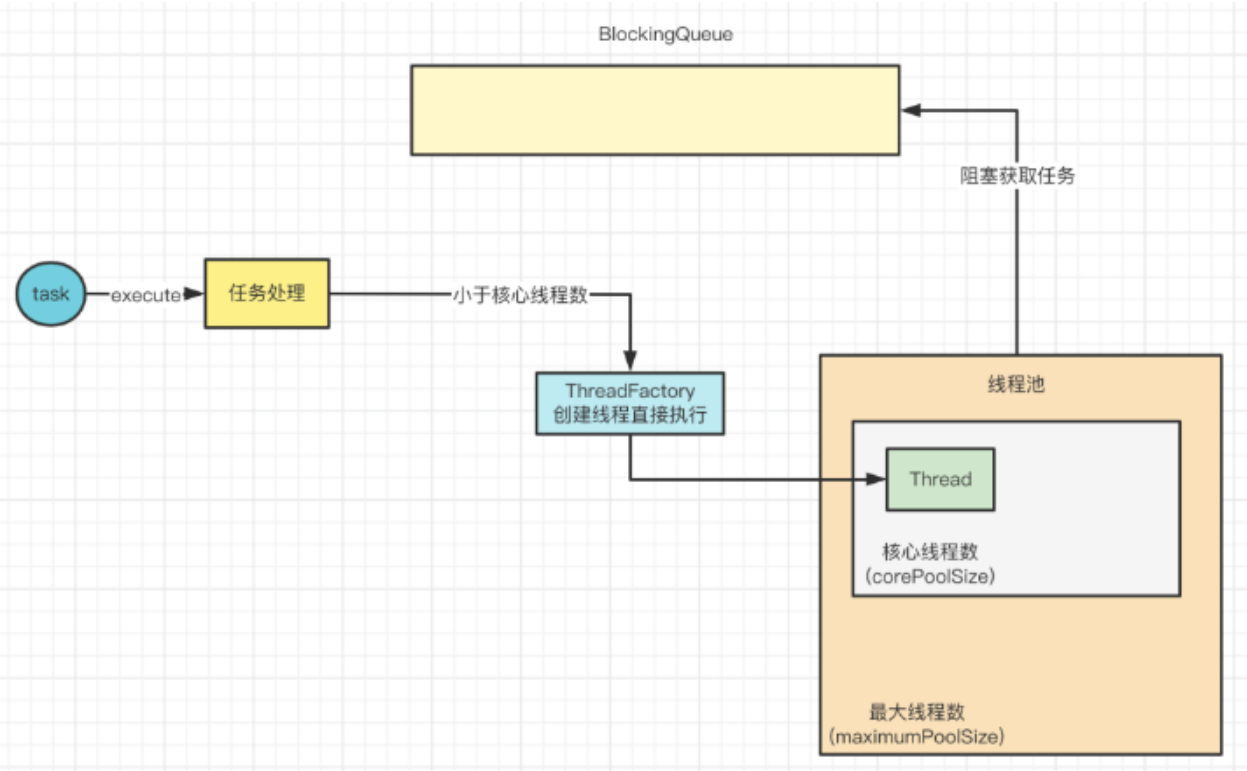
模型B.



3、线程池的运行原理

3.1、线程池的初始化

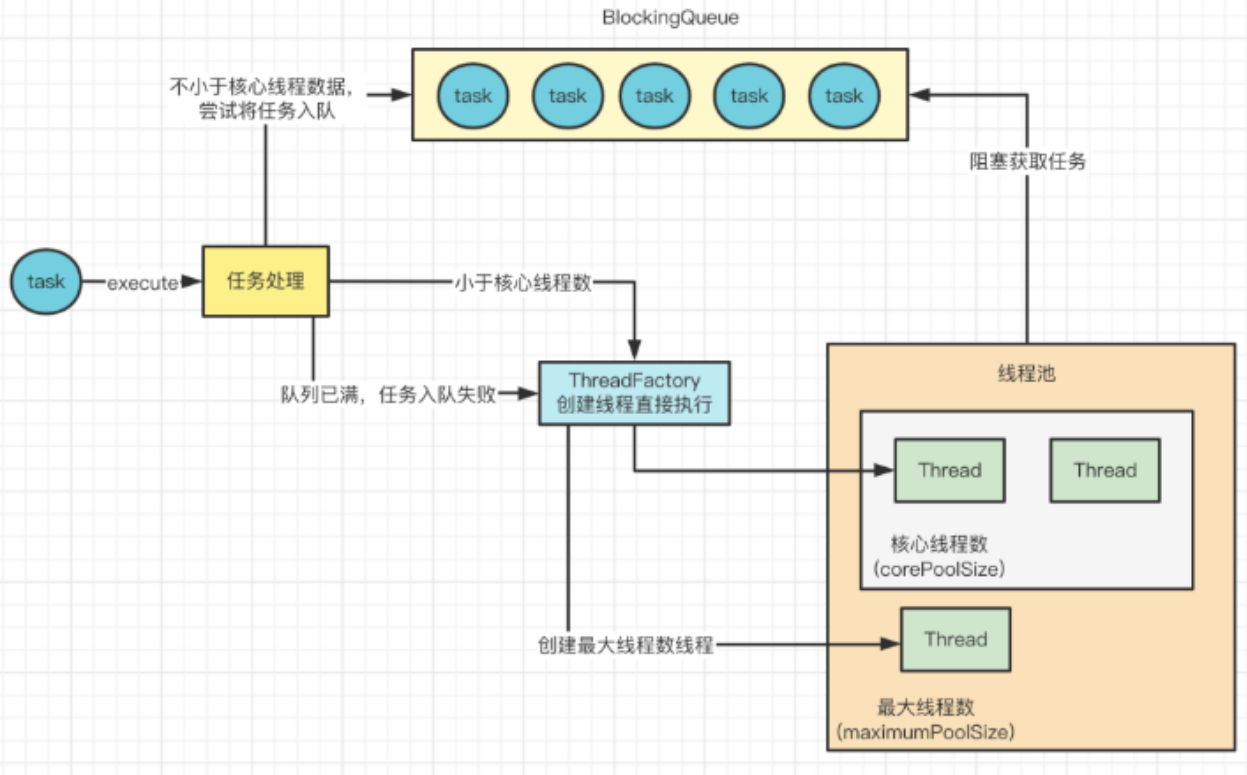
刚创建出来的线程池中只有一个构造时传入的阻塞队列而已，此时里面并没有的任何线程，但是如果你想要在执行之前已经创建好核心线程数，可以调用init方法来实现，默认是没有线程的.



3.2、扩大核心线程数

当有主线程通过addTask方法提交了一个任务，提交任务的时候，其实会去进行任务的处理

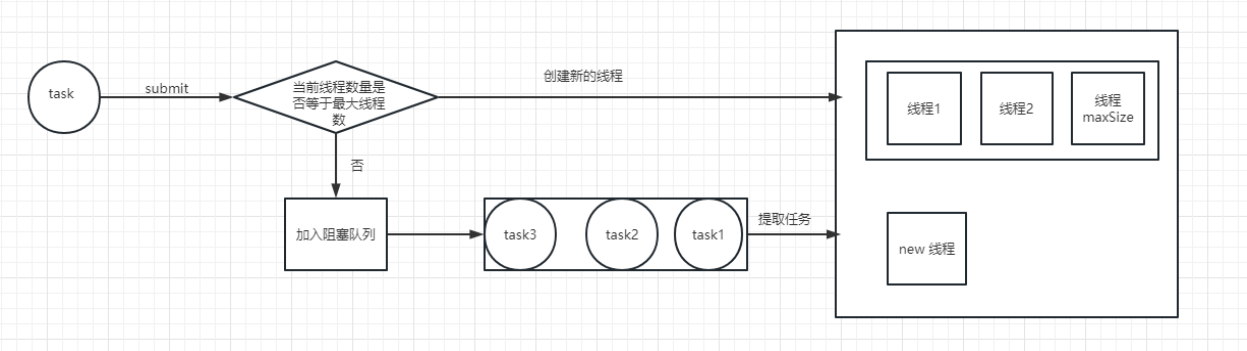
首先会去判断当前线程池的工作的线程数是否等于最大线程数。如果线程数已经达到了最大线程数量，那么就直接通过ThreadFactory创建一个线程来执行这个任务。



3.3、线程池的关闭

等待所有所有线程的退出，有一个没返回，就会继续等待。

二、完整流程图



Topic2: 进程池

一、什么是进程池

进程池中的所有子进程都运行相同的代码，有着相同的属性，因为进程池在服务器启动之初就创建好了，所有每个子进程都相对干净，没有打开不必要的文件描述符（从父进程继承而来），也不会错误地使用大块的堆内存（从父进程复制得到）。

1、进程池的好处

- 动态创建进程（或线程）是比较耗费时间的，这将导致较慢的客户响应。
- 动态创建的子进程（或子线程）通常只用来为一个客户服务（除非我们做特殊的处理），这将导致系统上产生大量的细微进程（或线程），进程（或线程）间的切换将消耗大量CPU时间。

2、进程池的模型

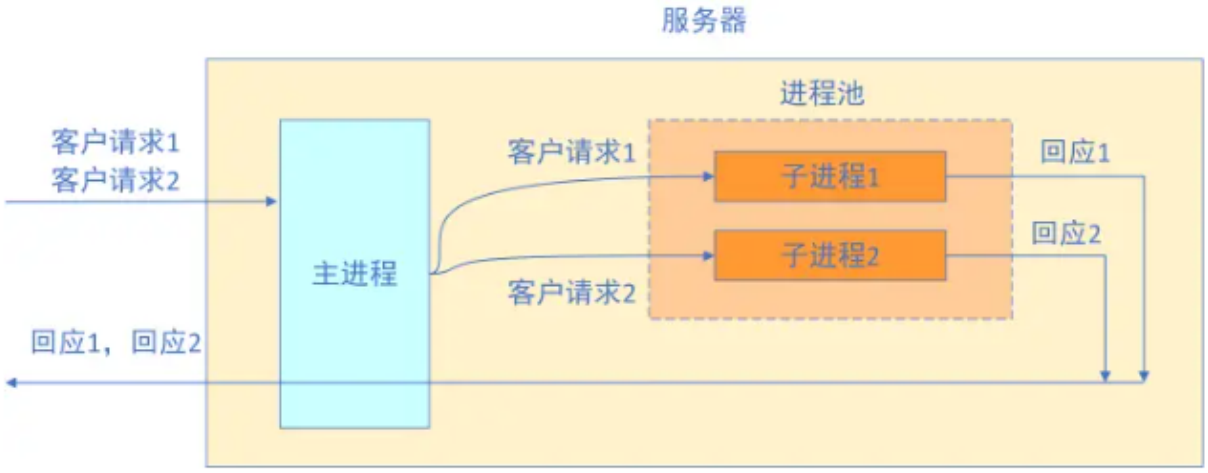
当新的任务来临，主进程选择哪个子进程来为新任务服务，则有两种方式： 1、主进程使用某种算法来主动选择子进程，最简单的算法是随机算法和Round Robin（轮流选取）算法。更加优秀的算法可以使任务在各个工作进程中更加均匀地分配，从而减轻服务器的整

体压力。 2、主线程和子线程通过一个共享的工作队列来同步，子进程都睡眠在该工作队列上。当有新任务来到时，主进程将任务添加到工作队列中。这将唤醒正在等待任务的子进程，不过只有一个子进程将获得信任的“接管权”，它可以从工作队列中取出任务并执行之，而其他子进程将继续睡眠在工作进程中。 当选择好子进程之后，主进程还需要使用某种通知机制来告诉目标子进程有新任务需要处理，并传递必要的数据。最简单的方法是，在父进程和子进程之间建立一条管道，通过管道来实现进程间通信。父线程和子线程之间的通信相对简单，只需要创建全局变量，因为它们共享全局变量。

综上所述，我们一般将进程池的一般模型描绘为下图所示。

2、进程池的模型

> 将进程和一个具体的cpu绑定到一起，解决缓存命中的问题
> 线程之间是共享内存的，一旦某个线程执行异常，内存出错，就会引起其他线程异常



nginx其实就是一个master进程和多个worker进程，master进程主要起监控的作用，比如有个worker进程挂了，就可以把worker进程拉起来，nginx热更新比如-s reload命令，就是把worker先全部干掉，再起新的进程。