第八章 线程池的使用

#### 一．在任务与执行策略之间的隐形耦合

1. 依赖性任务

2. 使用线程封闭机制的任务

3. 对响应时间敏感的任务

4. 使用ThreadLocal的任务

线程饥饿死锁

1. 线程池中的任务无限期的等待线程池中的其他任务提供的资源或条件。

2. 运行时间较长的任务会影响线程池的响应性。

#### 二．设置线程池的大小

1. Ncpu = CPU的数量

Ucpu = 目标CPU的使用率，0 ≤ Ucpu≤ 1

W/C = 等待时间与计算时间的比率

为保持处理器达到期望的使用率，最优的池的大小等于：

Nthreads = Ncpu \* Ucpu \* ( 1 + W/C )

可以使用Runtime来获得CPU的数目：

int N\_CPUS = Runtime.getRuntime().availableProcessors();

#### 三．配置ThreadPoolExecutor

线程的创建与销毁

1. public ThreadPoolExecutor(int corePoolSize,

int maximumPoolSize,

long keepAliveTime,

TimeUnit unit,

BlockingQueue<Runnable> workQueue,

RejectedExecutionHandler handler)

corePoolSize：线程池核心线程数量

maximumPoolSize:线程池最大线程数量

keepAliverTime：当活跃线程数大于核心线程数时，空闲的多余线程最大存活时间

unit：存活时间的单位

workQueue：存放任务的队列

handler：超出线程范围和队列容量的任务的处理程序

管理队列的任务

1. 无界队列

newFixedThreadPool和newSingleThreadPool在默认情况下将使用一个无界的LinkedBlockingQueue

2. 有界队列

3. 同步移交 SynchronousQueue

饱和策略

1. 当队列和线程池都满了，说明线程池处于饱和状态，那么必须对新提交的任务采用一种特殊的策略来进行处理。这个策略默认配置是AbortPolicy，表示无法处理新的任务而抛出异常。JAVA提供了4种策略：

（1）AbortPolicy：直接抛出异常

（2）CallerRunsPolicy：只用调用所在的线程运行任务

（3）DiscardOldestPolicy：丢弃队列里最近的一个任务，并执行当前任务

（4）DiscardPolicy：不处理，丢弃掉

线程工厂

1. ThreadFactory是一个线程工厂。用来创建线程。这里为什么要使用线程工厂呢？其实就是为了统一在创建线程时设置一些参数，如是否守护线程。线程一些特性等，如优先级。通过这个TreadFactory创建出来的线程能保证有相同的特性。它首先是一个接口类，而且方法只有一个。就是创建一个线程。

public interface ThreadFactory {

Thread newThread(Runnable r);

}

在JDK中，有实现ThreadFactory就只有一个地方。而更多的时候，我们都是继承它然后自己来写这个线程工厂的。

#### 四．扩展ThreadPoolExecutor

1. 扩展方法

（1）beforeExecutor

（2）afterExecutor

（3）terminated

#### 五．递归算法的并行化

1. 当串行的迭代操作彼此独立，并且每个迭代操作执行的工作量比管理一个新任务时带来的开销更多，那么这个串行循环就适合并行化。