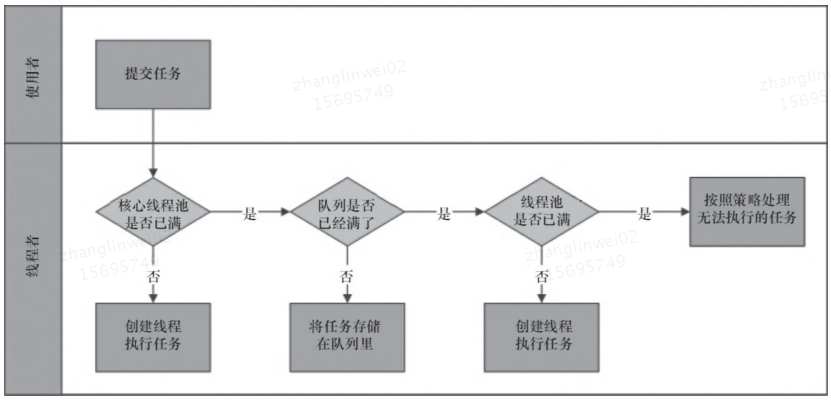
03 Java中的线程池

C-2 创建: 张林伟, 最后修改: 张林伟 04-01 10:58

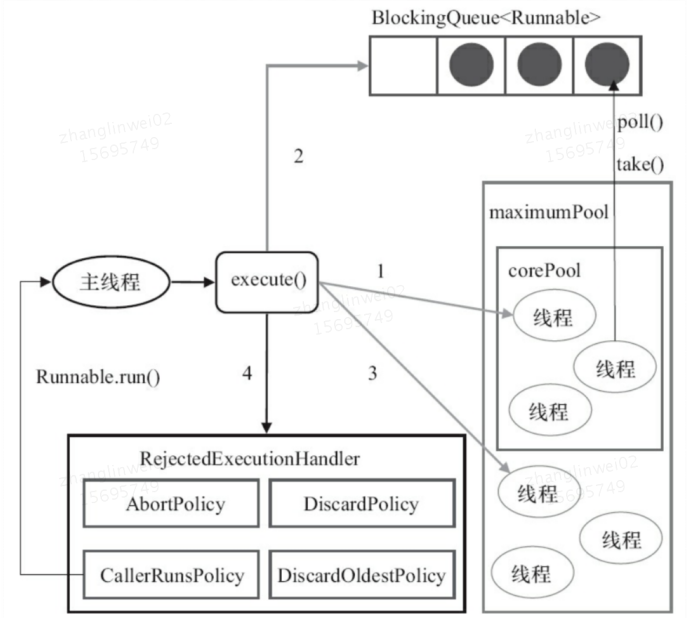
ThreadPoolExecutor线程池的主要处理流程



处理流程：

1. 当workerCount < corePoolSize，创建线程执行任务。
2. 当workerCount >= corePoolSize&&阻塞队列workQueue未满，把新的任务放入阻塞队列。
3. 当workQueue已满，并且workerCount >= corePoolSize，并且workerCount < maximumPoolSize，创建线程执行任务。
4. 当workQueue已满，workerCount >= maximumPoolSize，采取拒绝策略,默认拒绝策略是直接抛异常。

> 作者：Java架构技术栈
> 链接：https://juejin.im/post/5c9c98a85188252d8b13bc14
> 来源：掘金
> 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。



创建线程池的参数

- int corePoolSize
- int maximumPoolSize
- long keepAliveTime（线程活动保持时间：线程数大于corePoolSize时，线程池多余工作线程空闲后，保持存活的时间）
- TimeUnit unit（keepAliveTime的单位）
- BlockingQueue<Runnable> workQueue
  - ArrayBlockingQueue（基于数组、有界阻塞、FIFO）
  - LinkedBlockingQueue（基于链表、无界阻塞、FIFO）
  - SynchronousQueue（不存元素、每个插入操作必须等到另一个线程调用移除操作，否则一直阻塞）
  - PriorityBlockingQueue（优先级、无界阻塞）

- ThreadFactory threadFactory（可以通过线程工厂创建名称更有意义的线程）
- RejectedExecutionHandler handler

向线程池提交任务

- execute() 提交不需要返回值的任务
- submit() 提交需要返回指定的任务，返回 Future 类型的对象

关闭线程池

遍历线程池工作线程，调用线程的interrupt方法中断线程

- shutdown()
- showdownNow()

合理配置线程池

- CPU密集型：N_{cpu} + 1
- IO密集型：2 * N_{cpu}
- 混合型

ps.可以通过Runtime.getRuntime().availableProcessors()获取CPU核心数