

【Java多线程】Executor框架的详解

在Java中，使用线程来异步执行任务。Java线程的创建与销毁需要一定的开销。为每一个任务创建一个新线程来执行，这些线程的创建与销毁将消耗大量的计算资源。为每一个任务创建一个新线程来执行，这种策略可能会使处于高负荷状态的Java线程既是工作单元，也是执行单元。从JDK1.5开始，把工作单元与执行工作单元包括Runnable 和 Callable，而执行机制由Executor框架提供。

Executor框架简介

Executor框架的两级调度模型

1. Executor框架简介

1.1. Executor框架...

1.2. Executor框架...

1.2.1. Executor...

1.2.2. Executor...

1.2.3. Executor...

2. ThreadPoolE...

2.1. ThreadPoo...

2.2. 1. FixedT...

2.3. 2. SingleT...

2.4. 3. Cached...

告



明志健致远

5个月

)

2018年

<

9月

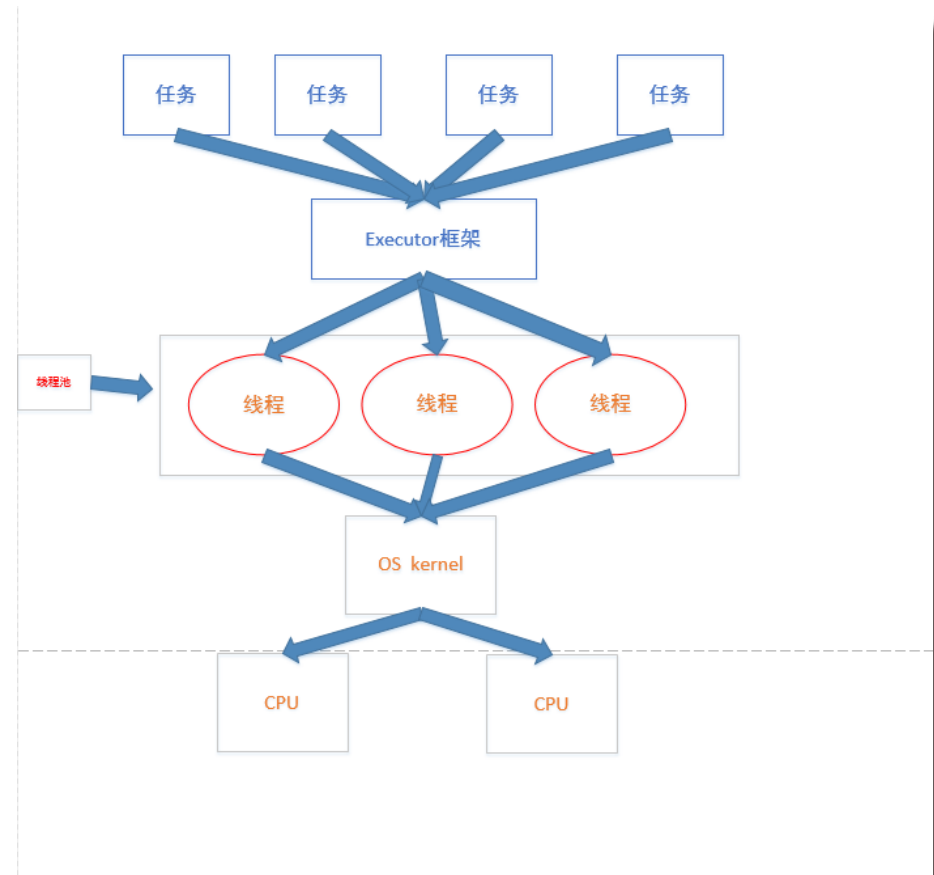
日 一 二 三 四

3
0

在HotSpot VM的线程模型中，Java线程被一对一映射为本地操作系统线程。Java线程启动时会创建一个本地操作系统线程；当Java线程终止时，这个操作系统线程也会被回收。操作系统会调用所有线程并将他们分配给可用的CPU。

可以将此种模式分为两层，在上层，Java多线程程序通常把应用程序分解为若干任务，然后使用用户级的调度器（Executor框架）将这些任务映射为固定数量的线程；在底层，操作系统内核将这些线程映射到硬件处理器上。

两级调度模型的示意图：



从图中可以看出，该框架用来控制应用程序的上层调度（下层调度由操作系统内核控制，不受应用程序的控制）。

26 27 28 29 30 31 1
2 3 4 5 6 7 8
 9 10 11 12 13 14 15
 16 17 18 19 20 21 22
 23 24 25 26 27 28 29
 30 1 2 3 4 5 6

标签

9)

1)

0)

.(8)

程(6)

eclipse(4)

Java集合(4)

3
0

Executor框架的结构和成员

Executor框架的结构

1. 任务

包括被执行任务需要实现的接口：Runnable接口和Callable接口

2. 任务的执行

包括任务执行机制的核心接口Executor，以及继承自Executor的ExecutorService接口。

Executor框架有两个关键类实现了ExecutorService接口：[ThreadPoolExecutor](#) 和 [ScheduledThreadPoolExecutor](#)

3. 异步计算的结果

包括Future和实现Future接口的FutureTask类。

Executor框架的类与接口

示意图

更多

积分与排名

积分 - 82167

排名 - 4771

最新评论

生哈希(has

:_bin仔

改。...

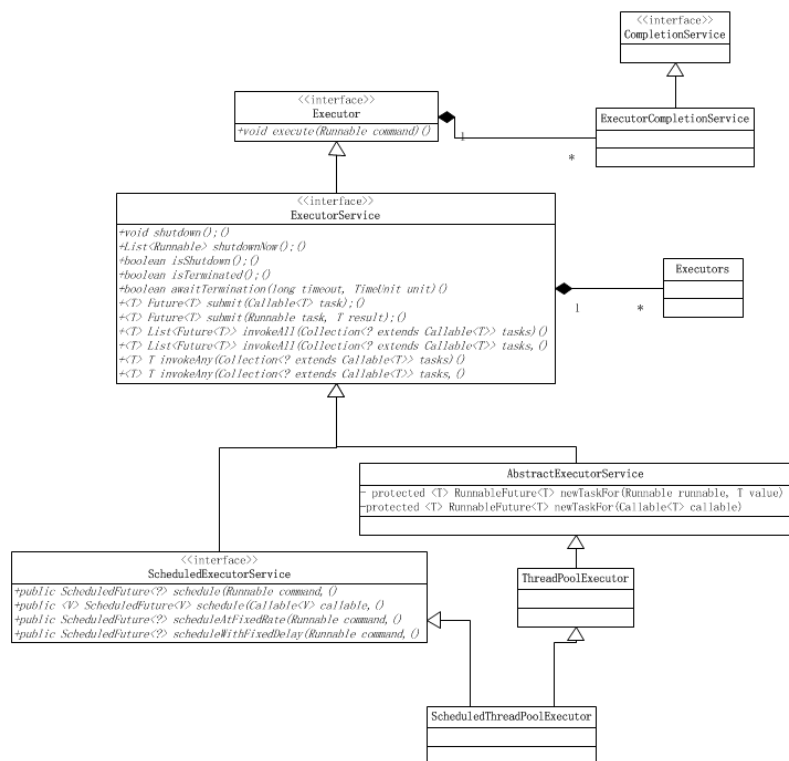
-明志健致远

JAVA程序员

享

评啊

!!!



Executor是一个接口，它是Executor框架的基础，它将任务的提交与任务的执行分离。
ThreadPoolExecutor是线程池的核心实现类，用来执行被提交的任务。

ScheduledThreadPoolExecutor是一个实现类，可以在给定的延迟后运行命令，或定期运行命令。**ScheduledThreadPoolExecutor**比**Timer**更灵活，功能更强大。

Future接口和它的实现**FutureTask**类，代表异步计算的结果。

Runnable和**Callable**接口的实现类，都可以被**ThreadPoolExecutor**或**ScheduledThreadPoolExecutor**执行。

Executor框架的使用

先来看个图：

--吃肉不长肉的小灏哥

3. Re:一致性哈希(hashed)算法

博主有地方写错了：

原文：...一个int

占8个字节，一个

long占16个字节，一个

double占8个字节，一个

float占4个字节，一个

short占2个字节，一个

byte占1个字节，一个

humor_bin

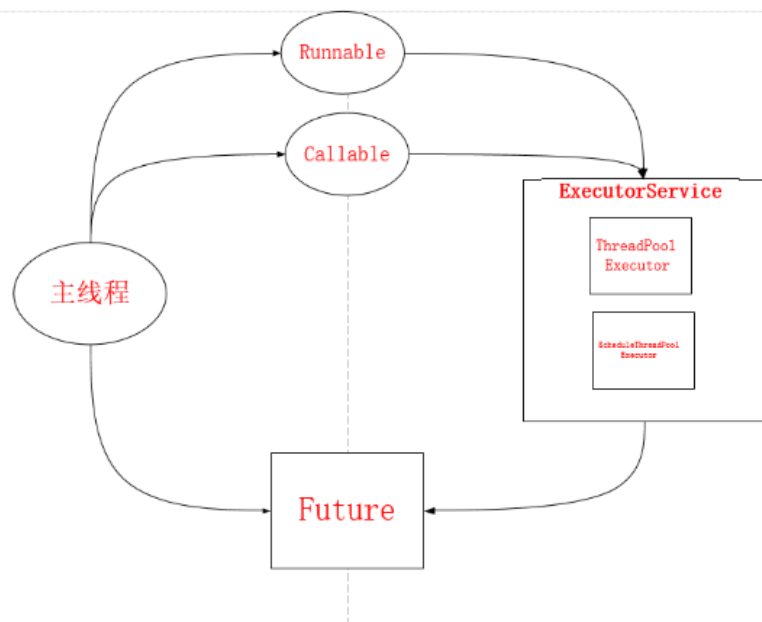
非行榜

新消息主...

Initializin...

3. 分布式系统ses 3

4. ConcurrentHa 0



1. 主线程首先要创建实现 Runnable接口或者Callable接口的任务对象。工具类一个Runnable对象封装为一个Callable对象

?

```
1 Executors.callable(Runnale task);
```

2或

```
3 Executors.callable(Runnable task, Object resule);
```

2. 然后可以把Runnable对象直接交给ExecutorService执行

?

```
1 ExecutorService1.execute(Runnable command);
```

```
2 或者也可以把Runnable对象或Callable对象提交给ExecutorService执行
```

```
3 ExecutorService.submit(Runnable task);
```

如果执行ExecutorService.submit(...), ExecutorService将返回一个实现Future接口的对象(到目前为止的JDK中, 返回的是FutureTask对象)。由于FutureTask实现了Runnable接口, 我们也可以创建FutureTask类, 然后直接交给ExecutorService执行。

5. spring + redis 实...

6. 【Java多线程】Ex...

7. 【Java基础】Java...

8. 2017年读书计划 (...)

9. 认证鉴权与API权...

10. Hibernate三种状...

11. 多线程技术: 两个...

12. 认证鉴权与API权...

正则表达...

充架构分...

什么, 怎...

七总结(23...

勺使用简...

自定义样...

发工具类...

竟配置为1....

j: "Previo...

以机 (一...

23. 并发编程中的

24. Dubbo超时和

3
0

3. 最后，主线程可以执行FutureTask.get()方法来等待任务执行完成。主线程也可以执行FutureTask.cancel(boolean mayInterruptIfRunning)来取消此任务的执行。

ThreadPoolExecutor详解

Executor框架最核心的类是[ThreadPoolExecutor](#)

ThreadPoolExecutor的组件构成

corePool：核心线程池的大小

maximumPool：最大线程池的大小

BlockingQueue：用来暂时保存任务的工作队列

RejectedExecutionHandler：当ThreadPoolExecutor已经关闭或ThreadPoolExecutor到了最大线程池的大小且工作队列已满，execute()方法将要调用的Handler。

Executor 可以创建3种类型的[ThreadPoolExecutor](#)线程池：

1. FixedThreadPool

创建固定长度的线程池，每次提交任务创建一个线程，直到达到线程池的最大大小不再变化。

这个线程池可以创建固定线程数的线程池。**特点就是可以重用固定数量线程**！构造源码如下：

?

```
1 public static ExecutorService newFixedThreadPool(int nThreads) {
2     return new ThreadPoolExecutor(nThreads, nThreads,
3     ...
4 }
```

25. spring源码学习...

26. HTTP 和 HTTPS...

27. eclipse maven 导...

28. 认证鉴权与API...

29. 一致性哈希(hash...

30. 记一次Spring的a...

评论排行榜

读书计划 (...)

【基础】Java...

使用简介...

定义样式...

希(hash)...

pring的ao...

加载的一...

HTTPS(3)

与API权...

正则表达...

11. Java虚拟机笔记 30

4

5 LinkedBlockingQueue<Runnable>());

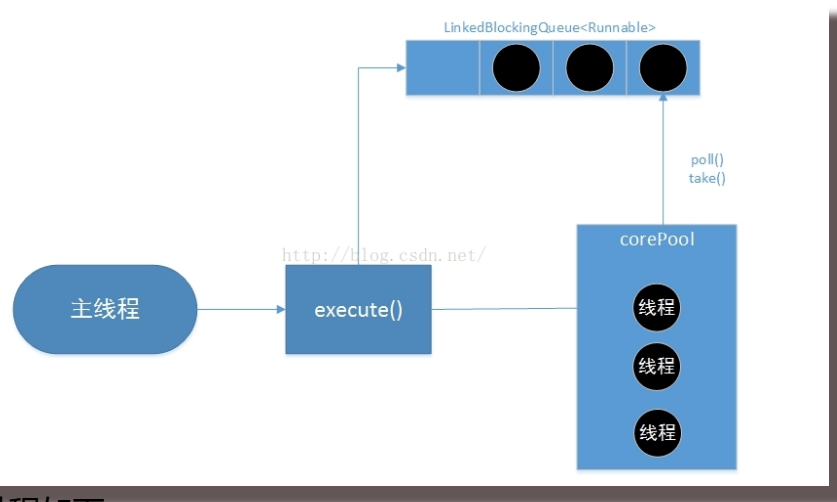
}

FixedThreadPool的corePoolSize和maxiumPoolSize都被设置为创建FixedThreadPool时指定的参数nThreads。

oL则表示当线程池中的线程数量操作核心线程的数量时，多余的线程将被立即停止

最后一个参数表示FixedThreadPool使用了无界队列LinkedBlockingQueue作为线程池的做工队列，由于是无界的，当线程池的线程数达到corePoolSize后，新任务将在无界队列中等待，因此线程池的线程数量不会超过corePoolSize，同时maxiumPoolSize也就变成了一个无效的参数，并且运行中的线程池并不会拒绝任务。

FixedThreadPool运行图如下



执行过程如下：

- 1.如果当前工作中的线程数量少于corePool的数量，就创建新的线程来执行任务
- 2.当线程池的工作中的线程数量达到了corePool，则将任务加入LinkedBlockingQueue。
- 3.线程执行完1中的任务后会从队列中去任务。

注意LinkedBlockingQueue是无界队列，所以可以一直添加新任务到线程池。

12. 一个JAVA程序员...

13. 分布式系统sessio...

14. 服务器有新消息...

15. eclipse maven 导...

16. ConcurrentHash...

17. 换了电脑如何使...

推荐排行榜

书计划 (...)

|定义样式...

!中的几个...

!新消息主...

entHash...

| HTTPS(6)

:则表达式...

【基础】Java...

!使用简介...

【多线程】E...

11. spring中Initia 3

0

2. SingleThreadExecutor

SingleThreadExecutor是使用单个worker线程的Executor。**特点是使用单个工作线程执行任务**。它的构造源码如下：

?

```
public static ExecutorService newSingleThreadExecutor() {
1         return new FinalizableDelegatedExecutorService
2             (new ThreadPoolExecutor(1, 1,
3                 TimeUnit.MILLISECONDS,
4                 new LinkedBlockingQueue<Runnable>()));
5     }
6 }
```

SingleThreadExecutor的corePoolSize和maxiumPoolSize都被设置1。

其他参数均与FixedThreadPool相同，其运行图如下：

12. Hibernate三种状...

13. TCP/IP三次握手...

14. 认证鉴权与API权...

15. 概述史：五胡十...

16. Java类的加载的...

17. JNDI是什么，怎...

18. SQL优化总结(1)

19. 多线程技术: 两个...

哈希(hash...

易景下的h...

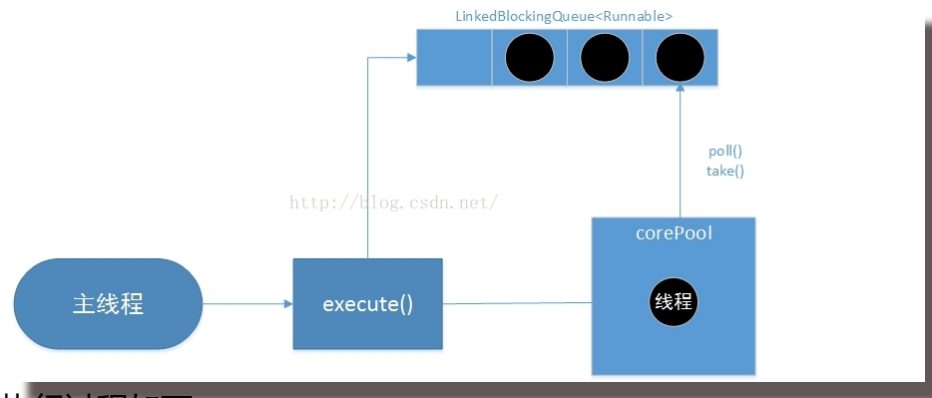
系统sessio...

录原理与...

权与API权...

权与API权...

3
0



执行过程如下：

- 1.如果当前工作中的线程数量少于corePool的数量，就创建一个新的线程来执行任务。
- 2.当线程池的工作中的线程数量达到了corePool，则将任务加入LinkedBlockingQueue。
- 3.线程执行完1中的任务后会从队列中去任务。

注意：由于在线程池中只有一个工作线程，所以任务可以按照添加顺序执行。

3. CachedThreadPool

CachedThreadPool是一个“无限”容量的线程池，它会根据需要创建新线程。

需要来创建新的线程执行任务，没有特定的corePool。下面是它的构造方法

?

```
1 public static ExecutorService newCachedThreadPool ()
```

```
2         return new ThreadPoolExecutor (0, 1
```

```
3
```

```
4 TimeUnit.SECONDS,
```

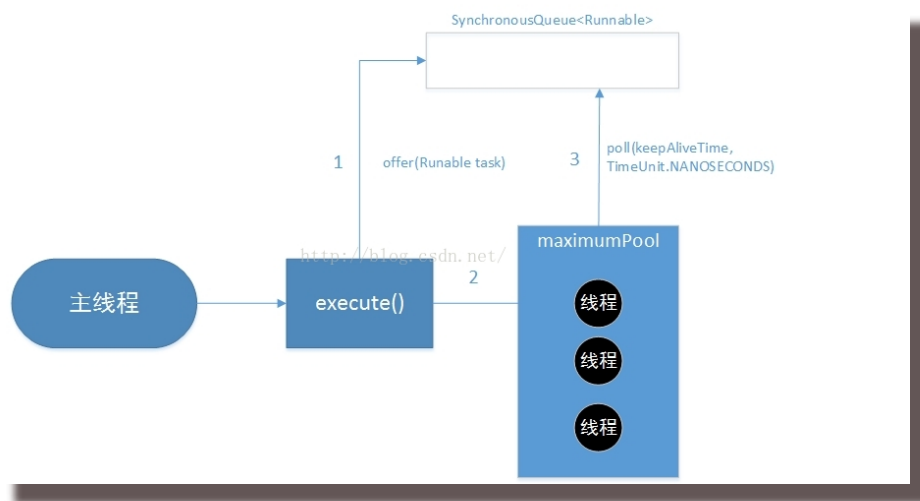
```
5
```

3
0

```
SynchronousQueue<Runnable>());  
}
```

CachedThreadPool的corePoolSize被设置为0，即corePool为空；maximumPoolSize被设置为Integer.MAX_VALUE，即maximum是无界的。这里keepAliveTime设置为60秒，意味着空闲的线程最多可以等待任务60秒，否则将被回收。

CachedThreadPool使用没有容量的SynchronousQueue作为主线程池的工作队列，量的阻塞队列。每个插入操作必须等待另一个线程的对应移除操作。这意味着，如果任务的速度高于线程池中处理任务的速度时，CachedThreadPool会不断创建新线程。CachedThreadPool会因为创建过多线程而耗尽CPU资源。其运行图如下：



3
0

执行过程如下：

- 1.首先执行SynchronousQueue.offer(Runnable task)。如果在当前的线程池中有空闲的线程正在执行SynchronousQueue.poll(), 那么主线程执行的offer操作与空闲线程执行的poll操作配对成功, 主线程把任务交给空闲线程执行。 , execute()方法执行成功, 否则执行步骤2
- 2.当线程池为空(初始maximumPool为空)或没有空闲线程时, 配对失败, 将没有线程执行SynchronousQueue.poll操作。这种情况下, 线程池会创建一个新的线程执行任务。
- 3.在创建完新的线程以后, 将会执行poll操作。当步骤2的线程执行完成后, 将等待60秒, 如果此时主线程提交了一个新任务, 那么这个空闲线程将执行新任务, 否则被回收。因此长时间不提交任务的CachedThreadPool不会占用系统资源。

SynchronousQueue是一个不存储元素阻塞队列, 每次要进行offer操作时必须等待poll操作, 否则不能继续添加元素。

参考书籍: 《Java并发编程的艺术》, 《Java并发编程实战》, 《Java高并》
更多内容: <http://www.cnblogs.com/study-everyday/>

♥ 作者: **明志健致远**

♠ 出处: <http://www.cnblogs.com/study-everyday/>

◆ 本文版权归作者和博客园共有, 欢迎转载, 但必须保留此段声明, 且在文章页面明显位置给出原文, 保留追究法律责任的权利。

♣ 本博客大多为学习笔记或读书笔记, 本文如对你有帮助, 请多多推荐下此文, 如有错误欢迎指正, 相互学习, 共同进步。

标签: Java并发编程, 多线程

好文要顶

关注我

收藏该文



« **上一篇**: 博客园样式代码留存

» **下一篇**: Dubbo的使用简介

posted @ 2017-04-20 10:35 明志健致远 阅读(5429) 评论(0) 编辑 收藏

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

Copyright ©2018 明志健

3
0