# SpringBoot中的线程池，你真的会用么?

好好学java Today

收录于话题
#Springboot 4 #Java后端 45

**今日推荐**
**代码对比工具，就用这7个!**
**在 IDEA 中的各种调试技巧，轻松定位 Bug（超级全面）**
**后端接口如何提高性能?**
**16 个写代码的好习惯**
**为什么不推荐使用BeanUtils属性转换工具**
**盘点阿里巴巴 34 个牛逼 GitHub 项目**

来源：*blog.csdn.net/m0_37701381/article/details/81072774*

## 前言

前两天做项目的时候，想提高一下插入表的性能优化，因为是两张表，先插旧的表，紧接着插新的表，一万多条数据就有点慢了

后面就想到了线程池ThreadPoolExecutor，而用的是Spring Boot项目，可以用Spring提供的对ThreadPoolExecutor封装的线程池ThreadPoolTaskExecutor，直接使用注解启用

## 使用步骤

先创建一个线程池的配置，让Spring Boot加载，用来定义如何创建一个ThreadPoolTaskExecutor，要使用@Configuration和@EnableAsync这两个注解，表示这是个配置类，并且是线程池的配置类

```
@Configuration
@EnableAsync
```

```
public class ExecutorConfig {

    private static final Logger logger = LoggerFactory.getLogger(ExecutorConfig.class);

    @Value("${async.executor.thread.core_pool_size}")
    private int corePoolSize;
    @Value("${async.executor.thread.max_pool_size}")
    private int maxPoolSize;
    @Value("${async.executor.thread.queue_capacity}")
    private int queueCapacity;
    @Value("${async.executor.thread.name.prefix}")
    private String namePrefix;

    @Bean(name = "asyncServiceExecutor")
    public Executor asyncServiceExecutor() {
        logger.info("start asyncServiceExecutor");
        ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
        //配置核心线程数
        executor.setCorePoolSize(corePoolSize);
        //配置最大线程数
        executor.setMaxPoolSize(maxPoolSize);
        //配置队列大小
        executor.setQueueCapacity(queueCapacity);
        //配置线程池中的线程的名称前缀
        executor.setThreadNamePrefix(namePrefix);

        // rejection-policy：当pool已经达到max size的时候，如何处理新任务
        // CALLER_RUNS：不在新线程中执行任务，而是有调用者所在的线程来执行
        executor.setRejectedExecutionHandler(new ThreadPoolExecutor.CallerRunsPolicy());
        //执行初始化
        executor.initialize();
        return executor;
    }
}
```

@Value是我配置在application.properties，可以参考配置，自由定义

```
# 异步线程配置
```

```
# 配置核心线程数
async.executor.thread.core_pool_size = 5
# 配置最大线程数
async.executor.thread.max_pool_size = 5
# 配置队列大小
async.executor.thread.queue_capacity = 99999
# 配置线程池中的线程的名称前缀
async.executor.thread.name.prefix = async-service-
```

创建一个Service接口，是异步线程的接口

```
public interface AsyncService {
    /**
     * 执行异步任务
     * 可以根据需求，自己加参数拟定，我这里就做个测试演示
     */
    void executeAsync();
}
```

实现类

```
@Service
public class AsyncServiceImpl implements AsyncService {
    private static final Logger logger = LoggerFactory.getLogger(AsyncServiceImpl.class);

    @Override
    @Async("asyncServiceExecutor")
    public void executeAsync() {
        logger.info("start executeAsync");

        System.out.println("异步线程要做的事情");
        System.out.println("可以在这里执行批量插入等耗时的事情");

        logger.info("end executeAsync");
    }
}
```

将Service层的服务异步化，在executeAsync()方法上增加注解@Async("asyncServiceExecutor")，asyncServiceExecutor方法是前面ExecutorConfig.java中的方法名，表明executeAsync方法进入的线程池是asyncServiceExecutor方法创建的。（搜索公众号Java知音，回复“2021”，送你一份Java面试题宝典）

接下来就是在Controller里或者是哪里通过注解@Autowired注入这个Service

```
@Autowired
private AsyncService asyncService;

@GetMapping("/async")
public void async(){
    asyncService.executeAsync();
}
```

用postmain或者其他工具来多次测试请求一下

```
 2018-07-16 22:15:47.655  INFO 10516 --- [async-service-5] c.u.d.e.executor.impl.AsyncServic
异步线程要做的事情
可以在这里执行批量插入等耗时的事情
2018-07-16 22:15:47.655  INFO 10516 --- [async-service-5] c.u.d.e.executor.impl.AsyncService
2018-07-16 22:15:47.770  INFO 10516 --- [async-service-1] c.u.d.e.executor.impl.AsyncService
异步线程要做的事情
可以在这里执行批量插入等耗时的事情
2018-07-16 22:15:47.770  INFO 10516 --- [async-service-1] c.u.d.e.executor.impl.AsyncService
2018-07-16 22:15:47.816  INFO 10516 --- [async-service-2] c.u.d.e.executor.impl.AsyncService
异步线程要做的事情
可以在这里执行批量插入等耗时的事情
2018-07-16 22:15:47.816  INFO 10516 --- [async-service-2] c.u.d.e.executor.impl.AsyncService
2018-07-16 22:15:48.833  INFO 10516 --- [async-service-3] c.u.d.e.executor.impl.AsyncService
异步线程要做的事情
可以在这里执行批量插入等耗时的事情
2018-07-16 22:15:48.834  INFO 10516 --- [async-service-3] c.u.d.e.executor.impl.AsyncService
2018-07-16 22:15:48.986  INFO 10516 --- [async-service-4] c.u.d.e.executor.impl.AsyncService
异步线程要做的事情
可以在这里执行批量插入等耗时的事情
2018-07-16 22:15:48.987  INFO 10516 --- [async-service-4] c.u.d.e.executor.impl.AsyncService
```

通过以上日志可以发现，[async-service-]是有多个线程的，显然已经在我们配置的线程池中执行了，并且每次请求中，controller的起始和结束日志都是连续打印的，表明每次请求都快速响应了，而耗时的操作都留给线程池中的线程去异步执行；

虽然我们已经用上了线程池，但是还不清楚线程池当时的情况，有多少线程在执行，多少在队列中等待呢？这里我创建了一个ThreadPoolTaskExecutor的子类，在每次提交线程的时候都会将当前线程池的运行状况打印出来

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;
import org.springframework.util.concurrent.ListenableFuture;

import java.util.concurrent.Callable;
import java.util.concurrent.Future;
import java.util.concurrent.ThreadPoolExecutor;

/**
 * @Author: ChenBin
 */
public class VisiableThreadPoolTaskExecutor extends ThreadPoolTaskExecutor {

    private static final Logger logger = LoggerFactory.getLogger(VisiableThreadPoolTaskExecu

    private void showThreadPoolInfo(String prefix) {
        ThreadPoolExecutor threadPoolExecutor = getThreadPoolExecutor();

        if (null == threadPoolExecutor) {
            return;
        }

        logger.info("{}, {},taskCount [{}], completedTaskCount [{}], activeCount [{}], queue
                this.getThreadNamePrefix(),
                prefix,
                threadPoolExecutor.getTaskCount(),
                threadPoolExecutor.getCompletedTaskCount(),
                threadPoolExecutor.getActiveCount(),
                threadPoolExecutor.getQueue().size());
    }
```

```
    @Override
    public void execute(Runnable task) {
        showThreadPoolInfo("1. do execute");
        super.execute(task);
    }

    @Override
    public void execute(Runnable task, long startTimeout) {
        showThreadPoolInfo("2. do execute");
        super.execute(task, startTimeout);
    }

    @Override
    public Future<?> submit(Runnable task) {
        showThreadPoolInfo("1. do submit");
        return super.submit(task);
    }

    @Override
    public <T> Future<T> submit(Callable<T> task) {
        showThreadPoolInfo("2. do submit");
        return super.submit(task);
    }

    @Override
    public ListenableFuture<?> submitListenable(Runnable task) {
        showThreadPoolInfo("1. do submitListenable");
        return super.submitListenable(task);
    }

    @Override
    public <T> ListenableFuture<T> submitListenable(Callable<T> task) {
        showThreadPoolInfo("2. do submitListenable");
        return super.submitListenable(task);
    }
}
```

如上所示，showThreadPoolInfo方法中将任务总数、已完成数、活跃线程数，队列大小都打印出来了，然后Override了父类的execute、submit等方法，在里面调用showThreadPoolInfo方法，这样每次有任务被提交到线程池的时候，都会将当前线程池的基本情况打印到日志中；

修改ExecutorConfig.java的asyncServiceExecutor方法，将ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor()改为ThreadPoolTaskExecutor executor = new VisiableThreadPoolTaskExecutor()

```
@Bean(name = "asyncServiceExecutor")
    public Executor asyncServiceExecutor() {
        logger.info("start asyncServiceExecutor");
        //在这里修改
        ThreadPoolTaskExecutor executor = new VisiableThreadPoolTaskExecutor();
        //配置核心线程数
        executor.setCorePoolSize(corePoolSize);
        //配置最大线程数
        executor.setMaxPoolSize(maxPoolSize);
        //配置队列大小
        executor.setQueueCapacity(queueCapacity);
        //配置线程池中的线程的名称前缀
        executor.setThreadNamePrefix(namePrefix);

        // rejection-policy：当pool已经达到max size的时候，如何处理新任务
        // CALLER_RUNS：不在新线程中执行任务，而是有调用者所在的线程来执行
        executor.setRejectedExecutionHandler(new ThreadPoolExecutor.CallerRunsPolicy());
        //执行初始化
        executor.initialize();
        return executor;
    }
```

再次启动该工程测试

```
2018-07-16 22:23:30.951  INFO 14088 --- [nio-8087-exec-2] u.d.e.e.i.VisiableThreadPoolTaskEx
2018-07-16 22:23:30.952  INFO 14088 --- [async-service-1] c.u.d.e.executor.impl.AsyncService
异步线程要做的事情
```

```
可以在这里执行批量插入等耗时的事情
2018-07-16 22:23:30.953  INFO 14088 --- [async-service-1] c.u.d.e.executor.impl.AsyncService
2018-07-16 22:23:31.351  INFO 14088 --- [nio-8087-exec-3] u.d.e.e.i.VisiableThreadPoolTaskE
2018-07-16 22:23:31.353  INFO 14088 --- [async-service-2] c.u.d.e.executor.impl.AsyncService
异步线程要做的事情
可以在这里执行批量插入等耗时的事情
2018-07-16 22:23:31.353  INFO 14088 --- [async-service-2] c.u.d.e.executor.impl.AsyncService
2018-07-16 22:23:31.927  INFO 14088 --- [nio-8087-exec-5] u.d.e.e.i.VisiableThreadPoolTaskE
2018-07-16 22:23:31.929  INFO 14088 --- [async-service-3] c.u.d.e.executor.impl.AsyncService
异步线程要做的事情
可以在这里执行批量插入等耗时的事情
2018-07-16 22:23:31.930  INFO 14088 --- [async-service-3] c.u.d.e.executor.impl.AsyncService
2018-07-16 22:23:32.496  INFO 14088 --- [nio-8087-exec-7] u.d.e.e.i.VisiableThreadPoolTaskE
2018-07-16 22:23:32.498  INFO 14088 --- [async-service-4] c.u.d.e.executor.impl.AsyncService
异步线程要做的事情
可以在这里执行批量插入等耗时的事情
2018-07-16 22:23:32.499  INFO 14088 --- [async-service-4] c.u.d.e.executor.impl.AsyncService
```

注意这一行日志：

```
2018-07-16 22:23:32.496  INFO 14088 --- [nio-8087-exec-7] u.d.e.e.i.VisiableThreadPoolTaskE
```

这说明提交任务到线程池的时候，调用的是submit(Callable task)这个方法，当前已经提交了3个任务，完成了3个，当前有0个线程在处理任务，还剩0个任务在队列中等待，线程池的基本情况一路了然；

**推荐文章**

- **14个项目！**
- **一款小清新的 SpringBoot+ Mybatis 前后端分离后台管理系统项目**
- **47K Star 的SpringBoot+MyBatis+docker电商项目，附带超详细的文档！**
- **写博客能月入10K？**
- **一款基于 Spring Boot 的现代化社区（论坛/问答/社交网络/博客）**

**更多项目源码**

- **这或许是最美的Vue+Element开源后台管理UI**
- **推荐一款高颜值的 Spring Boot 快速开发框架**
- **一款基于 Spring Boot 的现代化社区（论坛/问答/社交网络/博客）**
- **13K点赞都基于 Vue+Spring 前后端分离管理系统ELAdmin，大爱**
- **想接私活时薪再翻一倍，建议根据这几个开源的SpringBoot**

Read more

People who liked this content also liked

最常用的 Linux 命令都不会，你怎么敢去面试?

MarkerHub

---

图解 Spring 循环依赖，写得太好了!

SpringForAll社区

---

synchronized 的超多干货!

程序员cxuan