动态线程池组件

由 叶凯创建 不到1分钟以前

背景

在微服务的架构中，线程池的使用非常频繁，一般用来解决以下几个场景：

- 快速处理业务请求

  业务处理逻辑中包含对几个下游业务的调用，根据各自的返回结果做处理，使用线程池并发发起调用比串行调用性能更高；

- 主逻辑同步处理完成，部分逻辑独立异步处理，用以提升上游调用方的响应速度，这类异步处理可以直接扔进线程池，由线程池负责调度处理；
- 批量离线任务异步处理

目前用法

- 直接使用 jdk concurrent 包的 Exexcutors 工具类创建线程
- 使用 jdk concurent 包通过线程池构造函数创建 （new ...）

当前问题

- 线程池的运行是个黑盒
- 线程池的核心参数可能设置不合理，无法动态调整
- 线程池的任务队列无法监控，有可能任务生产过快，处理过慢，大量任务堆积在队列中，造成处理延时，甚至内存溢出（有实际案列）

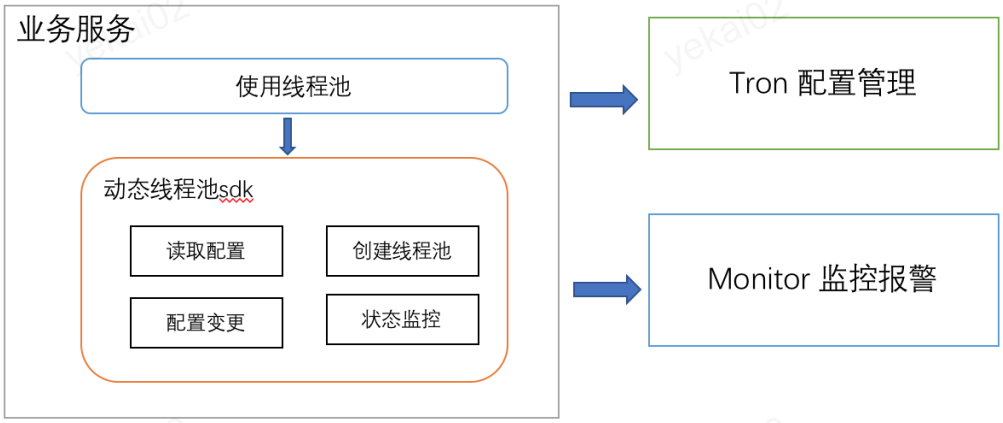
解决方案

- 尽量不用线程池？不可能，不用线程池无法享受其带来的好处，尤其是对于性能要求很高的系统线程池是必须的；
- 公式化线程池的配置参数？不现实；
- 动态线程池方案？支持核心参数动态调整，支持线程池的运行状态的实时监控，对线程池的创建使用统一化管理！

动态线程池（参考美团 https://tech.meituan.com/2020/04/02/java-pooling-pratice-in-meituan.html）

核心设计包括以下三个方面：

1. 简化线程池配置：配置最核心的基本参数，Less is More。
2. 参数可动态修改：为了解决参数不好配，修改参数成本高等问题。在Java线程池留有高扩展性的基础上，封装线程池，允许线程池监听同步外部的消息，根据消息进行修改配置。将线程池的配置放置在平台侧，允许开发同学简单的查看、修改线程池配置。
3. 增加线程池监控：对某事物缺乏状态的观测，就对其改进无从下手。在线程池执行任务的生命周期添加监控能力，帮助开发同学了解线程池状态。



配置

配置方式：目前支持在 tron dynamic-executors 项目下增加线程池动态配置，一个项目对应一个配置项，每组配置支持多个线程池

		项目	配置项	配置内容
☑	6885	dynamic-executors	sayHi-demo	[{"executorName":"sync-executor","coreSize":1,"maxSize":10,"keepAliveSeconds":120,"queueType":"LinkedBlockingQueue","queueSize":1000,"rejectHandler":"AbortPolicy","threadNamePrefix":"sync-","monitorCodes":["工作线程负荷","待执行的任务数","队列中的任务数","reject数量"]},{"executorName":"sayHi","coreSize":5,"maxSize":20,"keepAliveSeconds":120,"queueType":"LinkedBlockingQueue","queueSize":5000,"rejectHandler":"DiscardOl

配置说明：

```
[
    {
        "executorName":"sync-executor",//线程池名字，业务方根据名字获取对应的线程池
        "coreSize":1,//线程池核心线程数
        "maxSize":10,//最大线程数
        "keepAliveSeconds":120,//线程 keep alive 时间
        "queueType":"LinkedBlockingQueue",//任务队列类型，目前仅支持 "LinkedBlockingQueue", "
        "queueSize":1000,//队列大小，对于 LinkedBlockingQueue 有效
        "rejectHandler":"AbortPolicy",//任务拒绝策略，支持 jdk 自带的 "AbortPolicy", "CallerR
        "threadNamePrefix":"sync-",//线程名前缀，方便查看线程运行状态
        "monitorCodes":["工作线程负荷", "待执行的任务数", "队列中的任务数", "reject数量"] //对应四
    },
```

```
    {
        "executorName":"sayHi",
        "coreSize":5,
        "maxSize":20,
        "keepAliveSeconds":120,
        "queueType":"LinkedBlockingQueue",
        "queueSize":5000,
        "rejectHandler":"DiscardOldestPolicy",
        "threadNamePrefix":"sayHi-",
        "monitorCodes":["999095000000", "999095000001", "999095000002", "999095000003"]
    }
]
```

核心类 & Api

	说明	
xxx.xxx.executors.framework.ExecutorGroup	动态线程池组，维护根据线程池配置创建的所有线程池。 业务服务使用 ExecutorGroup.get(name) 方法获取对应的线程池	
xxx.xxxx.executors.framework.EnhancedThreadPoolExecutor	自定义的增强型线程池，增加监控、动态调参等辅助功能， 核心的任务执行、线程调度直接复用原有的线程池功能	
xxx.xxxx.executors.LocalConfigExecutorGroupFactory	基于本地文件创建 ExecutorGroup 的工厂类	
xxx.xxxx.executors.ZKConfigExecutorGroupFactory	基于 Tron (zk) 配置创建 ExecutorGroup 的工厂类	
xxx.xxxx.executors.DefaultExecutorGroupMonitor	默认的线程池状态监控类（上报到 monitor 服务），业务方也可以实现 ExecutorGroupMonitor 接口自定义监控实现	

业务服务使用样例

pom 依赖：

```
<dependency>
    <groupId>xxx.xxxx.dynamic-executors</groupId>
    <artifactId>dynamic-executors-client</artifactId>
    <version>0.0.1</version>
</dependency>
```

spring 配置：

```
<!--============================== Dynamic Executors 核心配置 ========================-->
```

```
    <!--============================LocalConfigExecutorGroupFactory====================
    <!-- <bean id="executorGroupFactory" class="xxx.xxxx.executors.LocalConfigExecutorGro
        <property name="localFile" value="executors.conf"/>
    </bean> -->

    <!--============================ZKConfigExecutorGroupFactory=======================
    <bean id="zkClient" class="xxx.xxxx.executors.zk.ExecutorsConfigZKClient" init-method
        <property name="connectionString" value="zk-server:port"/>
    </bean>
    <bean id="executorGroupFactory" class="xxx.xxxx.executors.ZKConfigExecutorGroupFactor
        <property name="appNodeName" value="sayHi-demo"/>
        <property name="zkClient" ref="zkClient"/>
    </bean>

    <bean id="executorGroup" factory-bean="executorGroupFactory" factory-method="create"/

    <bean id="executorGroupMonitor" class="xxx.xxxx.executors.DefaultExecutorGroupMonitor
        <property name="executorGroup" ref="executorGroup"/>
        <property name="monitorDataUploader" ref="monitorDataUploader"/>
    </bean>
```

code fragment

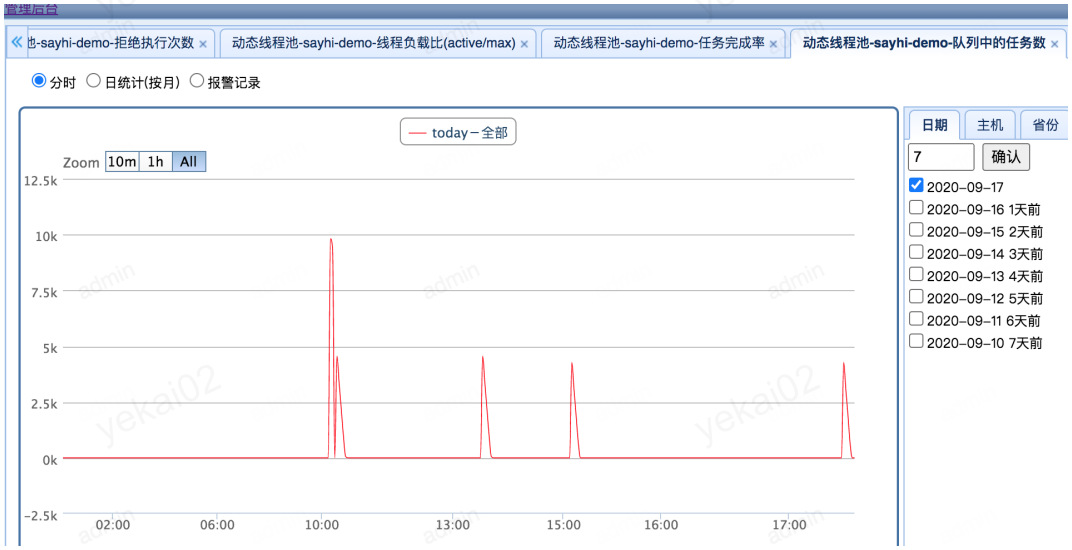
```
public class SayHiService {
    @Resource
    private ExecutorGroup executorGroup;

    private Executor executor;

    @PostConstruct
    public void init() {
        //executor = Executors.newFixedThreadPool(10); // Deperated !!!
        executor = executorGroup.get("sayHi");
        Assert.notNull(executor);
    }

    public void sayHi() {
        executor.execute(new Runnable() {
            @Override
            public void run() {
                System.out.println("hi");
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
    }
}
```

监控展示：


管理后台
池-sayhi-demo-拒绝执行次数
动态线程池-sayhi-demo-线程负载比(active/max)
动态线程池-sayhi-demo-任务完成率
动态线程池-sayhi-demo-队列中的任务数
分时
日统计(按月)
报警记录
today－全部
Zoom
10m
1h
All
12.5k
10k
7.5k
5k
2.5k
0k
-2.5k
02:00
06:00
10:00
13:00
15:00
16:00
17:00
日期
主机
省份
7
确认
2020-09-17
2020-09-16 1天前
2020-09-15 2天前
2020-09-14 3天前
2020-09-13 4天前
2020-09-12 5天前
2020-09-11 6天前
2020-09-10 7天前

无标签