NPException
码龄4年 暂无认证

465	1万+	2137	160万+	
原创	周排名	总排名	访问	等级

1万+	362	471	169	1396
积分	粉丝	获赞	评论	收藏

私信 关注

搜博主文章

分类专栏

java 97篇
java8 2篇
spring 12篇
springBoot 41篇
springCould 56篇
mybatis 7篇

最新评论

spring boot返回json数据和完美使用fastjs…
公众号：Java Pro: 一键三连，给你磕头 ..

Logback详解
侠骨留香: 👍

Mapper.xml详解
qq_42920672: 请问能把你的User也展示一下吗

Java SimpleDateFormat 中英文时间格式…
taogeyongwu: 谢谢大佬

springboot连接elasticsearch 报错failed t…
wph-01: 赞，版本问题，一直没找到！

最新文章

Centos7系统根分区空间小,home空间大。怎么删除home分区 增加到分区

使用stream流将list集合中某一BigDecimal字段求和

Logback详解

2021年 1篇	2020年 12篇
2019年 355篇	2018年 194篇

目录

Springboot2.x集成RabbitMQ实现消费者…
前言
yml配置

6.Springboot2.x集成RabbitMQ实现消费者限流，手动ack确认

原创 NPException 2020-02-13 13:50:36 1377 收藏 5 版权
分类专栏： # rabbitmq

Springboot2.x集成RabbitMQ实现消费者限流，手动ack确认

前言

我们在实际项目中，可能在mq中积累了成千上万的消息，如果我们不进行限流，当我们打开消费者的时候一下子成千上万的消息一下子冲击过来，可能会造成服务器宕机，或者业务出现严重漏洞，所以我们需要进行消费者限流。首先我的springboot版本，springBootVersion = '2.2.1.RELEASE'。其他版本配置差别都不大。

注意：在版本2.0之前的版本中，只有一种MessageListenerContainer—SimpleMessageListenerContainer;在2.0之后有第二个容器——DirectMessageListenerContainer，配置如下：

```
    listener:
    #Container where the RabbitMQ consumer dispatches messages to an invoker thread
      type: simple
      simple:
        acknowledge-mode: manual
        prefetch: 1
    #Container where the listener is invoked directly on the RabbitMQ consumer thre
      type: direct
      direct:
        acknowledge-mode: manual
        prefetch: 1
```

SimpleMessageListenerContainer

默认情况下，侦听器容器将启动单个使用者，该使用者将从队列接收消息。

在检查上一节中的表时，您将看到许多控制并发性的属性。最简单的是concurrentConsumers，它只创建(固定的)将并发处理消息的使用者数量。

此外，还添加了一个新的属性maxConcurrentConsumers，容器将根据工作负载动态调整并发性。这与四个附加属性一起工作:continutiveactivetrigger、startConsumerMinInterval、continutiveidletrigger、stopConsumerMinInterval。

在默认设置下，增加消费者的算法工作如下:

如果尚未到达maxConcurrentConsumers，并且已有的使用者连续10个周期处于活动状态，并且自上一个使用者启动以来至少已经过了10秒，那么将启动一个新的使用者。如果使用者在txSize *中接收到至少一条消息，则认为该使用者处于活动状态。

在默认设置下，减少消费者的算法工作如下:

如果有多个concurrentConsumers正在运行，并且某个consumer检测到10个连续超时(空闲)，并且上一个consumer至少在60秒之前停止，那么该consumer将停止。超时取决于receiveTimeout和txSize属性。如果使用者在txSize *中没有接收到任何消息，则认为它是空闲的。因此，在默认超时(1秒)和txSize为4的情况下，在40秒的空闲时间(4个超时对应1个空闲检测)之后将考虑停止使用者。

DirectMessageListenerContainer

使用DirectMessageListenerContainer，您需要确保ConnectionFactory配置了一个任务执行器，该执行器在使用该ConnectionFactory的所有侦听器容器中具有足够的线程来支持所需的并发性。默认连接池大小仅为5。

并发性基于配置的队列和consumersPerQueue。每个队列的每个使用者使用一个单独的通道，并发性由rabbit客户端库控制;默认情况下，它使用5个线程池;您可以配置taskExecutor来提供所需的最大并发性。

对比

SimpleMessageListenerContainer提供了以下特性，但DirectMessageListenerContainer不提供:

- txSize—使用SimpleMessageListenerContainer，您可以将其设置为控制事务中传递的消息数量和/或减少ack的数量，但这可能会导致失败后重复传递的数量增加。(与txSize和SimpleMessageListenerContainer一样，DirectMessageListenerContainer也有mesagesPerAck，可以用来减少ack，但不能用于事务—每个消息都在单独的事务中交付和打包)。
- maxconcurrentconsumer和consumer伸缩间隔/触发器—DirectMessageListenerContainer中没有自动伸缩;但是，它允许您以编程方式更改consumersPerQueue属性，并相应地调整使用者。

然而，与SimpleMessageListenerContainer相比，DirectMessageListenerContainer有以下优点:

- 在运行时添加和删除队列更有效;使用SimpleMessageListenerContainer，整个使用者线程重新启动(所有使用者取消并重新创建);对于DirectMessageListenerContainer，不受影响的使用者不会被取消。
  避免了RabbitMQ客户机线程和使用者线程之间的上下文切换。
- 线程是跨使用者共享的，而不是为SimpleMessageListenerContainer中的每个使用者都有一个专用线程。但是，请参阅"线程和异步使用者"一节中有关连接工厂配置的重要说明。

yml配置

```
spring:
  application:
    name: zoo-plus-rabbitmq
  rabbitmq:
    virtual-host: /
    host: localhost
    port: 5672
    username: guest
    password: guest
    listener:
      type: simple
      simple:
        acknowledge-mode: manual #采用手动应答
        prefetch: 1 #限制每次发送一条数据。
```

新建一个队列:

```
    /**
     * 测试队列
     */
    @Bean
    public Queue testQueue() {
        return QueueBuilder.nonDurable("test-queue").build();
    }
```

生产者:

```

    /**
     * 发送五条数据，测试消费端必须ack才发送第二条，消费者限流
     */
    @GetMapping("ack")
    public Resp testAck() {
        for (int i = 0; i < 5; i++) {
            rabbitTemplate.convertAndSend("test-queue", "测试ack确认模式");
        }
        return Resp.success("ok", null);
    }
```

消费者:

```
    @RabbitListener(queues = {"test-queue"})
    public void testQueue(Message message, Channel channel) throws IOException {
        log.info("test-queue消费: " + new String(message.getBody()));

        /*
              listener:
                type: simple
                simple:
                  #采用手动应答
                  acknowledge-mode: manual
                  prefetch: 1 #限制每次发送一条数据。
         */
//         采用手动ack，一条条的消费
        channel.basicAck(message.getMessageProperties().getDeliveryTag(), false);

        /*
         * 还可以nack
         * 第三个参数是否重回队列
         */
//        channel.basicNack(message.getMessageProperties().getDeliveryTag(),false,t
    }
```

由上面测试，如果我们没有配置，一下子五条信息就全部拥挤上来了，如果我们加了配置，不在消费端进行ack的话，消息就会unacked，下次重启服务也会再一次发送这条消息，直到消费端ack。

源码地址：https://gitee.com/zoo-plus/springboot-learn/tree/2.x/springboot-middleware/rabbitmq