链滴

SpringCloud Alibaba 微服务实战二十二 - 整合 Dubbo

作者：jianzh5

原文链接：https://ld246.com/article/1607392819616

来源网站：链滴

概述

在Spring Cloud构建的微服务系统中，大多数的开发者使用都是官方提供的Feign组件来进行内部服通信，这种声明式的HTTP客户端使用起来非常的简洁、方便、优雅，但是有一点，在使用Feign消费务的时候，相比较Dubbo这种RPC框架而言，性能较差。

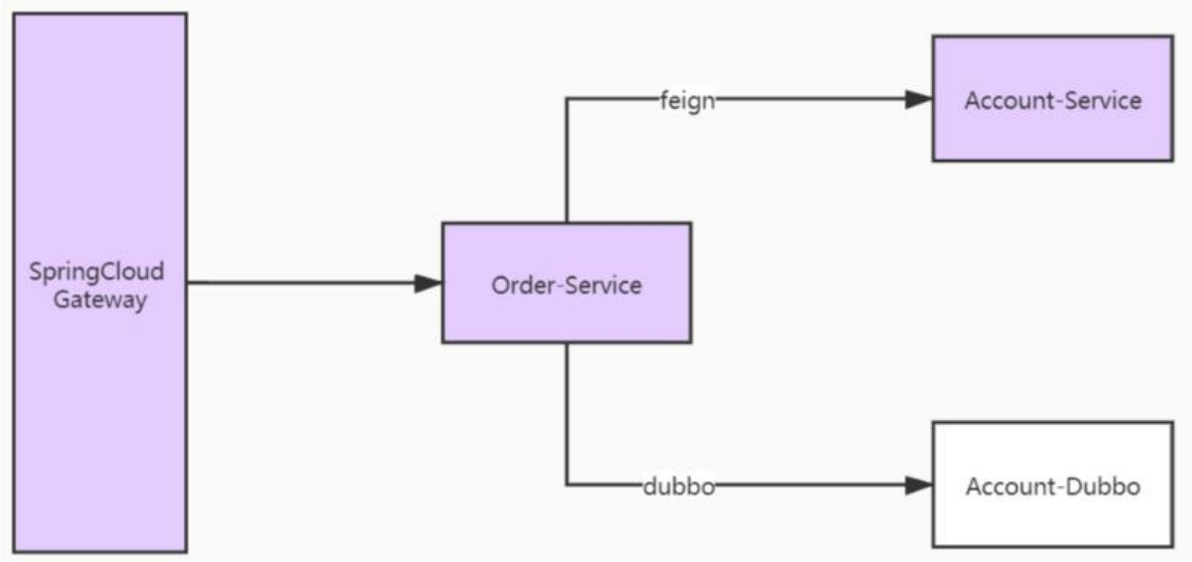
虽说在微服务架构中，会讲按照业务划分的微服务独立部署，并且运行在各自的进程中。微服务之间通信更加倾向于使用HTTP这种简答的通信机制，大多数情况都会使用REST API。这种通信方式非常简洁高效，并且和开发平台、语言无关，但是通常情况下，HTTP并不会开启KeepAlive功能，即当前接为短连接，短连接的缺点是每次请求都需要建立TCP连接，这使得其效率变的相当低下。

对外部提供REST API服务是一件非常好的事情，但是如果内部调用也是使用HTTP调用方式，就会显显得性能低下，Spring Cloud默认使用的Feign组件进行内部服务调用就是使用的HTTP协议进行调，这时，我们如果内部服务使用RPC调用，对外使用REST API，将会是一个非常不错的选择，恰巧，ubbo Spring Cloud给了我们这种选择的实现方式。

以上内容摘自官网。

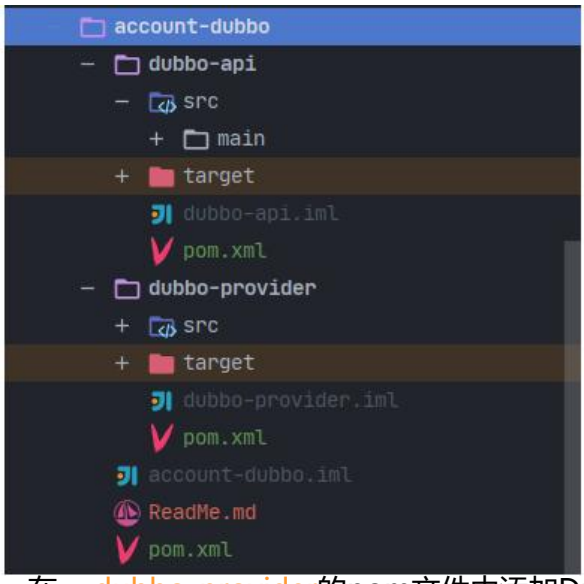
这里我觉得还有另外一个点也是需要大家考虑的，很多公司在早期已经基于dubbo构建了很多底层微务，如果全部使用cloud架构进行迁移相对比较麻烦，如果在cloud中能直接集成dubbo就简单多了。

之前的文章我们已经实现了从Order-service使用feign调用Account-Serviice，本章内容我们来实现用dubbo调用Account-Dubbo，最后通过Jmeter对接口进行性能测试看看两者之间的性能差距。



实现Dubbo生产者

● 在项目中建立服务模块 account-dubbo，在 account-dubbo中再建立两个模块 dubbo-api和 dbbo-provider 模块。



- 在 dubbo-provider的pom文件中添加Dubbo Spring Cloud的关键依赖。

```
<dependency>
   <groupId>com.alibaba.cloud</groupId>
   <artifactId>spring-cloud-starter-dubbo</artifactId>
</dependency>

<dependency>
   <groupId>com.alibaba.cloud</groupId>
   <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
</dependency>
```

注意：这里不能直接使用dubbo的原生jar，并且需要引入nacos注册中心，需要将dubbo注册到springcloud的注册中心上

- 修改 dubbo-provider 项目的配置文件application.properties

```
spring.cloud.nacos.discovery.server-addr = 10.0.23.48:8848
dubbo.application.id = account-dubbo
dubbo.protocol.name = dubbo
dubbo.protocol.port = 20881
dubbo.scan.base-packages = com.javadaily.dubbo.service
dubbo.registry.address = spring-cloud://localhost
spring.application.name = account-dubbo
dubbo.registry.check = false
dubbo.consumer.check = false
dubbo.cloud.subscribed-services = order-service
```

这里使用 dubbo.registry.address将dubbo注册到了springcloud的注册中心上。

- 在 dubbo-api中添加服务接口，为了后面的性能测试我们这里使用Account-Service一样的接口。

```
public interface AccountService {
   AccountDTO getByCode(String accountCode);
}
```

- 在 dubbo-provider中实现该接口（mybatis的相关配置略...）

```
@Service
public class AccountServiceImpl implements AccountService {

    @Autowired
    private AccountMapper accountMapper;

    @Override
    public AccountDTO getByCode(String accountCode) {
        AccountDTO accountDTO = new AccountDTO();
        Account account = accountMapper.selectByCode(accountCode);
        BeanUtils.copyProperties(account,accountDTO);
        return accountDTO;
    }
}
```

注意这里的 @Service注解使用的是 org.apache.dubbo.config.annotation.Service，不是Spring原的Service注解。

- 在 dubbo-provider中添加主启动类，并添加EnableDubbo注解

```
@SpringBootApplication
@EnableDubbo
public class AccountDubboApplication {
    public static void main(String[] args) {
        SpringApplication.run(AccountDubboApplication.class, args);
    }
}
```

- 启动dubbo服务，查看注册中心中是否正常注册。

Order-Service消费方修改

- 引入dubbo相关依赖

```
<!--dubbo-->
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-dubbo</artifactId>
</dependency>


<dependency>
    <groupId>com.jianzh5.cloud</groupId>
    <artifactId>dubbo-api</artifactId>
    <version>1.0.0</version>
</dependency>
```

● 修改消费方配置文件，增加dubbo的配置参数

```
dubbo:
  registry:
    address: spring-cloud://localhost
  cloud:
    subscribed-services: account-dubbo
  consumer:
    check: false
```

由于 Order-Service不需要对外提供RPC服务所以这里只需要配置dubbo的注册中心即可。

通过 dubbo.cloud.subscribed-services 配置生成者的服务id，默认是 * 即注册所有，此选项建议配
。

● 修改消费逻辑，使用dubbo调用。

```
@Slf4j
@RestController
public class FeignController {

    @Reference
    private AccountService accountService;

    @GetMapping("/order/getAccount/{accountCode}")
    public ResultData<AccountDTO> getAccount(@PathVariable String accountCode){

        AccountDTO accountDTO = accountService.getByCode(accountCode);

        return ResultData.success(accountDTO);
    }

}
```

通过 org.apache.dubbo.config.annotation.Reference 引入dubbo服务即可。

通过以上几步我们完成了SpringCloud与Dubbo的整合，大家自行测试即可。接下来我们看看两者之
的性能差异。

性能测试

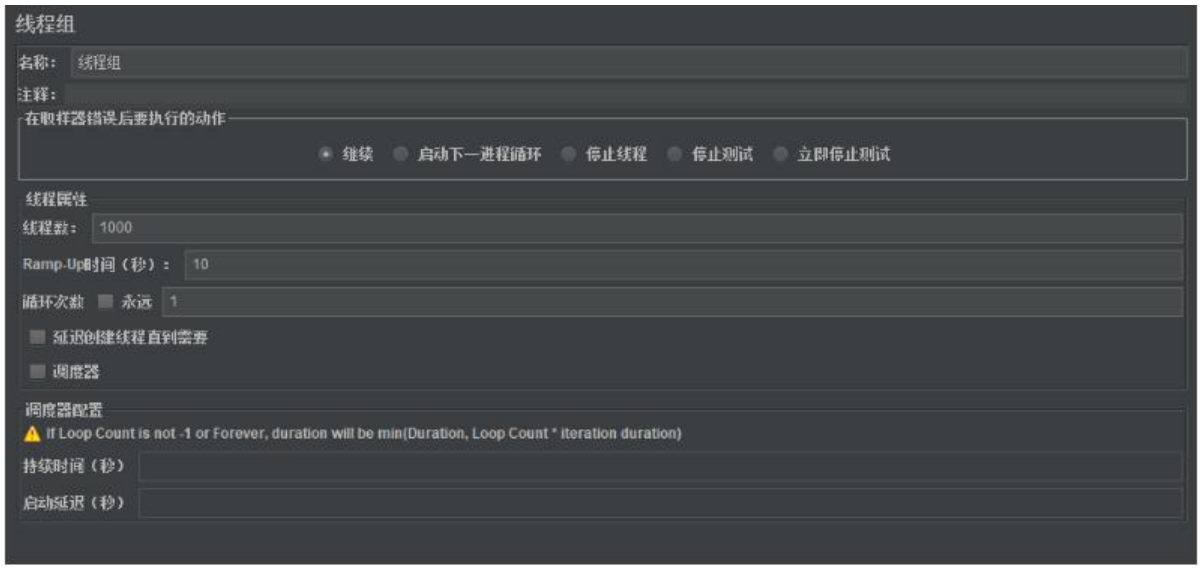
测试效果是在楼主笔记本上测试的，在高配置上效果差异会更明显。

配置：Intel Core i5-7200U CPU @ 2.50GHz 2.70GHz 6核 16G内存

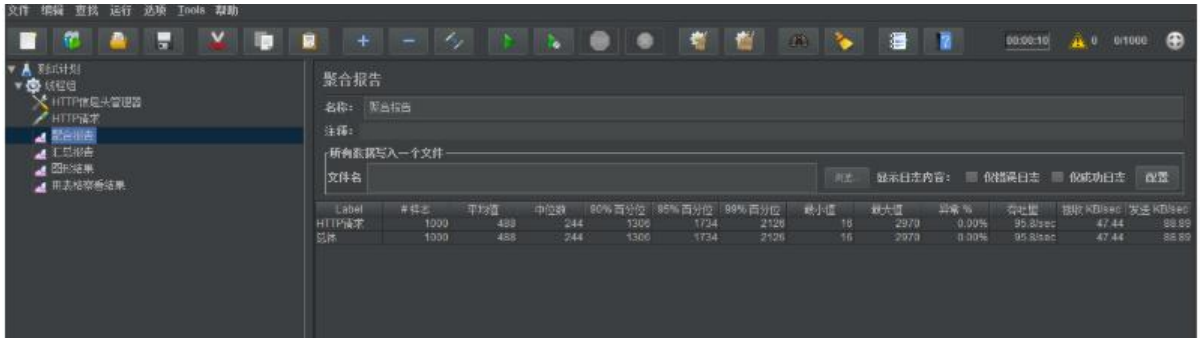
测试工具： JMeter5.1

线程数：1000

Ramp-Up : 10

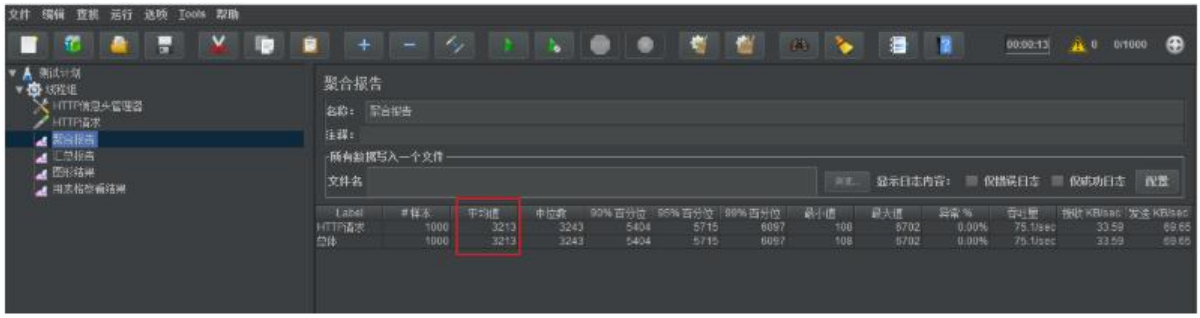


dubbo调用效果



平均响应时间	吞吐量	最小响应时间	最大响应时间
488ms	95.8/sec	16ms	2070ms

feign调用效果



平均响应时间	吞吐量	最小响应时间	最大响应时间
3213ms	75.1/sec	108ms	6097ms

通过上面的测试结果可以看出两者的性能差距，feign的性能明显落后dubbo很多，对于追求性能的

用来说还是建议使用dubbo进行RPC调用。