负载均衡策略与配置细节

Dubbo 支持的消费端负载均衡策略及其使用方法

Dubbo 内置了 client-based 负载均衡机制，如下是当前支持的负载均衡算法，结合上文提到的自动服务发现机制，消费端会自动使用 `Weighted Random LoadBalance 加权随机负载均衡策略` 选址调用。

如果要调整负载均衡算法，以下是 Dubbo 框架内置的负载均衡策略：

算法	特性	备注	配置值
Weighted Random LoadBalance	加权随机	默认算法，默认权重相同	random (默认)
RoundRobin LoadBalance	加权轮询	借鉴于 Nginx 的平滑加权轮询算法，默认权重相同	roundrobin
LeastActive LoadBalance	最少活跃优先 + 加权随机	背后是能者多劳的思想	leastactive
Shortest-Response LoadBalance	最短响应优先 + 加权随机	更加关注响应速度	shortestresponse
ConsistentHash LoadBalance	一致性哈希	确定的入参，确定的提供者，适用于有状态请求	consistenthash
P2C LoadBalance	Power of Two Choice	随机选择两个节点后，继续选择“连接数”较小的那个节点。	p2c
Adaptive LoadBalance	自适应负载均衡	在 P2C 算法基础上，选择二者中 load 最小的那个节	adaptive

全局配置

Dubbo 框架的默认策略是 `random` 加权随机负载均衡。如果要调整策略，只需要设置 `loadbalance` 相应取值即可，每种负载均衡策略取值请参见文档最上方表格。

为所有服务调用指定全局配置：

```
dubbo:
  consumer:
    loadbalance: roundrobin
```

接口级配置

可以为每个服务指定不同的负载均衡策略。

在 provider 端设置，作为 consumer 侧默认值

```
@DubboService(loadbalance = "roundrobin")
public class DemoServiceImpl implements DemoService {}
```

在 consumer 端设置，具备更高优先级

```
@DubboReference(loadbalance = "roundrobin")
private DemoService demoService;
```

方法级配置

也可以指定方法(method)级别的负载均衡策略。

在 Spring Boot 开发模式下，配置 method 级别参数有以下几种方式：

JavaConfig

```
    @Bean
    public ServiceBean demoService() {
        MethodConfig method = new MethodConfig();
        method.setName("sayHello");
        method.setLoadbalance("roundrobin");

        ServiceBean service = new ServiceBean();
        service.setInterface(DemoService.class);
        service.setRef(new DemoServiceImpl());
        service.addMethod(method)
        return service;
    }
}
```

```
@Autowire
private DemoService demoService;

@Configuration
public class DubboConfiguration {
    @Bean
    public ReferenceBean demoService() {
        MethodConfig method = new MethodConfig();
        method.setName("sayHello");
        method.setLoadbalance("roundrobin");

        ReferenceBean<DemoService> reference = new Referenc
        reference.setInterface(DemoService.class);
        reference.addMethod(method);
        return reference;
    }
}
```

dubbo.properties

```
dubbo.reference.org.apache.dubbo.samples.api.DemoService.sa
```

一致性哈希配置

默认采用第一个参数作为哈希 key，如果需要切换参数，可以指定 `hash.arguments` 属性

```
Map<String, String> parameters = new HashMap<String, String
parameters.put("hash.arguments", "1");
parameters.put("sayHello.hash.arguments", "0,1");
referenceConfig.setParameters(parameters);
referenceConfig.setLoadBalance("consistenthash");
referenceConfig.get();
```

自适应负载均衡配置

只需要在 consumer 或 provider 端将 `loadbalance` 设置为 `p2c` 或者 `adaptive` 即可，可在此查看 工作原理

反馈

此页是否对您有帮助?

是 否

最后修改 September 13, 2024: Refactor website structure (#2860) (1a4b998f54b)

快速开始

开发者指南