Spring Cloud Ribbon 负载均衡

负载均衡算法基础

资料：https://en.wikipedia.org/wiki/Load_balancing_(computing)

Tomcat :

IP: 127.0.0.1

PORT: 8080

简单负载均衡算法

- 随机：random choice
- 轮训：round robin
- 最小活跃：least connections（LRU）

复杂负载均衡算法（多因子 Factors）

- 权重因子
- 负载情况（CPU、Load）
- 响应时间（Response Time RT）

> Netflix Ribbon 一套实现相对简单（或薄弱）客户端的负载均衡框架

Ribbon 负载均衡算法

核心接口 - com.netflix.loadbalancer.IRule

服务实例的不同实现类

- Netflix Eureka - InstanceInfo
- Spring Cloud Commons：ServiceInstance
- Netflix Ribbon - Server

IRule 实现类

- 随机 - RandomRule
- 轮训 - RoundRobinRule
- 最可用 - BestAvailableRule
- 粘性 - StickyRule
- 权重+响应时间 - WeightedResponseTimeRule

负载均衡器实现 - com.netflix.loadbalancer.ILoadBalancer

ILoadBalancer 没有直接注入 IRule 接口（方法）

抽象实现 - AbstractLoadBalancer 同样也没有

直到具体实现 DynamicServerListLoadBalancer 才与 IRule 关联 - 构造器注入

ILoadBalancer 实现类是不负责直接控制负载均衡策略，而是通过外部注入 IRule 来控制，不过通过 chooseServer 来选择其中服务

ILoadBalancer 实现类相对较少，IRule 实现较多

- ILoadBalancer
  - AbstractLoadBalancer
    - BaseLoadBalancer

以 DynamicServerListLoadBalancer 为例：

> Dynamic 动态更新
>
> - 删除
> - 增加
>
> ServerList - 多台服务实例
>
> LoadBalancer - LoadBalancer 实现
>
> ILoadBalancer : IRule = 1 对 1

配置 - IClientConfig

负责规则 - IRule

判断指定服务实例是否存活 - IPing

服务实例列表 - ServerList

服务实例列表过滤 - ServerListFilter（过滤不可用服务）

- 总数十台机器
  - 过滤不可用机器（两台）
  - 蓝绿发布（蓝、绿）
  - 动态路由（北京地区、上海地区、杭州地区）

服务实例列表更新 - ServerListUpdater

- 客户端周期性更新 - 30s 一次拉取注册中心的服务
- 服务端推送 - 推送新增或者移除的服务实例，通知客户端更新

服务实例列表接口 - com.netflix.loadbalancer.ServerList

ServerList - 多台服务实例

动态列表 - 注册中心 - 服务发现（通过服务名称发现一或更多服务实例）

- Netflix Eureka - com.netflix.niws.loadbalancer.DiscoveryEnabledNIWSServerList
- Spring Cloud
  - Zookeeper - org.springframework.cloud.zookeeper.discovery.ZookeeperServerList
  - Consul - org.springframework.cloud.consul.discovery.ConsulServerList
  - Spring Cloud DiscoveryClient 实现

静态列表 - 配置服务实例

- com.netflix.loadbalancer.ConfigurationBasedServerList

服务实例存活接口 - com.netflix.loadbalancer.IPing

永远存活实现 - com.netflix.loadbalancer.DummyPing

没有实现 -com.netflix.loadbalancer.NoOpPing

可调整实现 - com.netflix.loadbalancer.PingConstant

远程状态获取实现 - com.netflix.loadbalancer.PingUrl

基于 Eureka 注册中心实现 - com.netflix.niws.loadbalancer.NIWSDiscoveryPing

基于 Consul 注册中心实现 - org.springframework.cloud.consul.discovery.ConsulPing

服务实例列表过滤接口 - com.netflix.loadbalancer.ServerListFilter

服务实例列表更新接口 - com.netflix.loadbalancer.ServerListUpdater

周期性轮训更新实现 - com.netflix.loadbalancer.PollingServerListUpdater

> Spring Cloud 采用默认实现

更新周期：30（默认） - 更新 Ribbon ServerList 缓存

> 如果 Ribbon 整合 Eureka 的话，请问服务列表更新周期是多久？
>
> Spring Cloud Eureka 配置 registryFetchIntervalSeconds = 30 秒
>
> Spring Cloud Eureka 服务注册周期时间 instanceInfoReplicationIntervalSeconds = 30 秒
>
> 最理想情况：0秒（注册时间）+ 30 秒（更新时间）
>
> 最恶劣情况：30秒（注册时间）+ 30秒（更新时间）
>
> 整合 Ribbon 后：
>
> 最理想情况：0秒（注册时间）+ 30 秒（更新时间）+ 0秒（更新 ServerList ）= 30 秒
>
> 最恶劣情况：30秒（注册时间）+ 30秒（更新时间）+ 30秒（更新 ServerList ）= 90 秒
>
> Spring Cloud 服务调用时服务实例发现周期 [30~90] 之间

> Spring Cloud 服务注册、发现、负载均衡使用三个线程：
>
> - Eureka CacheRefreshThread 线程
> - Eureka InstanceInfoReplicator 线程
> - Ribbon PollingServerListUpdater 线程

> 思考：Zookeeper Watch 监听机制，那么是否可以在此基础上做一些实时性变化实现

扩展议题：@Qualifier 在 Spring Framework 使用场景

`@LoadBalanced` 是 `@Qualfier` 注解的“派生”

`@Qualifier` 注解的双重含义：

- 指定 `value()` 属性 - 按照 Bean 名称（Name）或者别名（Alias）
- 如果不指定 `value()` 属性 - 按照 `@Qualfier` 元注解过滤

QualifierAnnotationAutowireCandidateResolver

org.springframework.beans.factory.support.DefaultListableBeanFactory#doResolveDependency