Spring Cloud Netflix Ribbon

Eureka 高可用架构

高可用注册中心集群

只需要增加 Eureka 服务器注册URL：

```
## Eureka Server 服务 URL,用于客户端注册
eureka.client.serviceUrl.defaultZone=\
  http://localhost:9090/eureka,http://localhost:9091/eureka
```

如果 Eureka 客户端应用配置多个 Eureka 注册服务器，那么默认情况只有第一台**可用**的服务器，存在注册信息。

如果 第一台**可用**的 Eureka 服务器 Down 掉了，那么 Eureka 客户端应用将会选择下一台**可用**的 Eureka 服务器。

配置源码（EurekaClientConfigBean）

配置项 `eureka.client.serviceUrl` 实际映射的字段为 `serviceUrl` `，它是 Map 类型，Key 为自定义，默认值“defaultZone”，value 是需要配置的Eureka 注册服务器URL。

```
private Map<String, String> serviceUrl = new HashMap<>();

{
  this.serviceUrl.put(DEFAULT_ZONE, DEFAULT_URL);
}
```

value 可以是多值字段，通过“,” 分割：

```
String serviceUrls = this.serviceUrl.get(myZone);
if (serviceUrls == null || serviceUrls.isEmpty()) {
   serviceUrls = this.serviceUrl.get(DEFAULT_ZONE);
}
if (!StringUtils.isEmpty(serviceUrls)) {
   final String[] serviceUrlsSplit =
StringUtils.commaDelimitedListToStringArray(serviceUrls);
}
```

获取注册信息时间间隔

Eureka 客户端需要获取 Eureka 服务器注册信息，这个方便服务调用。

Eureka 客户端：`EurekaClient`,关联应用集合：`Applications`

单个应用信息：`Application`，关联多个应用实例

单个应用实例：`InstanceInfo`

当 Eureka 客户端需要调用具体某个服务时，比如`user-service-consumer` 调用`user-service-provider`， `user-service-provider`实际对应对象是`Application`,关联了许多应用实例(`InstanceInfo`)。

如果应用`user-service-provider`的应用实例发生变化时，那么`user-service-consumer`是需要感知的。比如：`user-service-provider`机器从10 台降到了5台，那么，作为调用方的`user-service-consumer`需要知道这个变化情况。可是这个变化过程，可能存在一定的延迟，可以通过调整注册信息时间间隔来减少错误。

具体配置项

```
## 调整注册信息的获取周期，默认值：30秒
eureka.client.registryFetchIntervalSeconds = 5
```

实例信息复制时间间隔

具体就是客户端信息的上报到 Eureka 服务器时间。当 Eureka 客户端应用上报的频率越频繁，那么 Eureka 服务器的应用状态管理一致性就越高。

具体配置项

```
## 调整客户端应用状态信息上报的周期
eureka.client.instanceInfoReplicationIntervalSeconds = 5
```

> Eureka 的应用信息获取的方式：拉模式
>
> Eureka 的应用信息上报的方式：推模式

实例Id

从 Eureka Server Dashboard 里面可以看到具体某个应用中的实例信息，比如：

```
UP (2) - 192.168.1.103:user-service-provider:7075 , 192.168.1.103:user-service-provider:7078
```

其中，它们命名模式：`${hostname}:${spring.application.name}:${server.port}`

实例类：`EurekaInstanceConfigBean`

配置项

```
## Eureka 应用实例的ID
eureka.instance.instanceId = ${spring.application.name}:${server.port}
```

实例端点映射

源码位置：`EurekaInstanceConfigBean`

```
private String statusPageUrlPath = "/info";
```

配置项

```
## Eureka 客户端应用实例状态 URL
eureka.instance.statusPageUrlPath = /health
```

Eureka服务端高可用

构建 Eureka 服务器相互注册

Eureka Server 1 -> Profile : peer1

配置项

```
### Eureka Server 应用名称
spring.application.name = spring-cloud-eureka-server
### Eureka Server 服务端口
server.port= 9090
### 取消服务器自我注册
eureka.client.register-with-eureka=true
### 注册中心的服务器，没有必要再去检索服务
eureka.client.fetch-registry = true
## Eureka Server 服务 URL,用于客户端注册
## 当前 Eureka 服务器 向 9091（Eureka 服务器） 复制数据
eureka.client.serviceUrl.defaultZone=\
  http://localhost:9091/eureka
```

Eureka Server 2 -> Profile : peer2

配置项

```
### Eureka Server 应用名称
spring.application.name = spring-cloud-eureka-server
### Eureka Server 服务端口
server.port= 9091
### 取消服务器自我注册
eureka.client.register-with-eureka=true
### 注册中心的服务器，没有必要再去检索服务
eureka.client.fetch-registry = true
## Eureka Server 服务 URL,用于客户端注册
## 当前 Eureka 服务器 向 9090（Eureka 服务器） 复制数据
eureka.client.serviceUrl.defaultZone=\
  http://localhost:9090/eureka
```

通过`--spring.profiles.active=peer1`和`--spring.profiles.active=peer2`分别激活 Eureka Server 1 和 Eureka Server 2

Eureka Server 1 里面的replicas 信息：

registered-replicas	http://localhost:9091/eureka/

Eureka Server 2 里面的replicas 信息：

registered-replicas	http://localhost:9090/eureka/

Spring RestTemplate

HTTP消息装换器：HttpMessageConvertor

自义定实现

编码问题

切换序列化/反序列化协议

HTTP Client 适配工厂：ClientHttpRequestFactory

这个方面主要考虑大家的使用 HttpClient 偏好：

- Spring 实现
  - SimpleClientHttpRequestFactory
- HttpClient
  - HttpComponentsClientHttpRequestFactory
- OkHttp
  - OkHttp3ClientHttpRequestFactory
  - OkHttpClientHttpRequestFactory

举例说明

```
RestTemplate restTemplate = new RestTemplate(new
HttpComponentsClientHttpRequestFactory()); // HTTP Client
```

切换HTTP 通讯实现，提升性能

HTTP 请求拦截器：ClientHttpRequestInterceptor

加深RestTemplate 拦截过程的

整合Netflix Ribbon

`RestTemplate`增加一个`LoadBalancerInterceptor`，调用Netflix 中的`LoadBalander`实现，根据 Eureka 客户端应用获取目标应用 IP+Port 信息，轮训的方式调用。

实际请求客户端

- LoadBalancerClient
  - RibbonLoadBalancerClient

负载均衡上下文

- LoadBalancerContext
  - RibbonLoadBalancerContext

负载均衡器

- ILoadBalancer
  - BaseLoadBalancer
  - DynamicServerListLoadBalancer
  - ZoneAwareLoadBalancer
  - NoOpLoadBalancer

负载均衡规则

核心规则接口

- IRule
  - 随机规则：RandomRule
  - 最可用规则：BestAvailableRule
  - 轮训规则：RoundRobinRule
  - 重试实现：RetryRule
  - 客户端配置：ClientConfigEnabledRoundRobinRule
  - 可用性过滤规则：AvailabilityFilteringRule
  - RT权重规则：WeightedResponseTimeRule
  - 规避区域规则：ZoneAvoidanceRule

PING 策略

核心策略接口

- IPingStrategy

PING 接口

- IPing
  - NoOpPing
  - DummyPing
  - PingConstant
  - PingUrl

Discovery Client 实现

- NIWSDiscoveryPing

问答部分

1. 为什么要用eureka?

   答：目前业界比较稳定云计算的开发员中间件，虽然有一些不足，基本上可用
2. 使用的话-每个服务api都需要eureka 插件

   答：需要使用 Eureka 客户端
3. eureka 主要功能为啥不能用浮动ip 代替呢?

   答：如果要使用浮动的IP，也是可以的，不过需要扩展
4. 这节内容是不是用eureka 来解释 负载均衡原理、转发规则计算?

   答：是的
5. eureka 可以替换为 zookeeper和consoul 那么这几个使用有什么差异?

   答：https://www.consul.io/intro/vs/zookeeper.html

   https://www.consul.io/intro/vs/eureka.html
6. 通讯不是指注册到defaultZone配置的那个么?

   答：默认情况是往 defaultZone 注册
7. 如果服务注册中心都挂了,服务还是能够运行吧?

   答：服务调用还是可以运行，有可能数据会不及时、不一致
8. spring cloud 日志收集 有解决方案么?

   答：一般用 HBase、或者 TSDB、elk
9. spring cloud提供了链路跟踪的方法吗

   答：http://cloud.spring.io/spring-cloud-static/Dalston.SR4/single/spring-cloud.html#_spring_cloud_sleuth