crazyhzm 265

Dubbo源码解析（三十四）集群——开篇

java dubbo 阅读约 12 分钟

集群——开篇

> 目标：介绍接下来集群分哪几部分来描述，介绍dubbo在集群中涉及到的几个功能，介绍dubbo-cluster下跟各个功能相关的接口

集群是什么？

如果说分布式是爸爸住在杭州，妈妈住在上海，那么集群就是两个爸爸，一个住在杭州，一个住在上海。对于分布式和集群有高吞吐量、高可用的目标。对于分布式来说每个服务器部署的服务任务是不同的，可能需要这些服务器上的服务共同协作才能完成整个业务流程，各个服务各司其职。而集群不一样，集群是同一个服务，被部署在了多个服务器上，每个服务器的任务都是一样的，是为了减少压力集中的问题，而集群中就会出现负载均衡、容错等问题。

dubbo的集群涉及到以下几部分内容：

1. 目录：Directory可以看成是多个Invoker的集合，但是它的值会随着注册中心中服务变化推送而动态变化，那么Invoker以及如何动态变化就是一个重点内容。
2. 集群容错：Cluster 将 Directory 中的多个 Invoker 伪装成一个 Invoker，对上层透明，伪装过程包含了容错逻辑，调用失败后，重试另一个。
3. 路由：dubbo路由规则，路由规则决定了一次dubbo服务调用的目标服务器，路由规则分两种：条件路由规则和脚本路由规则，并且支持可拓展。
4. 负载均衡策略：dubbo支持的所有负载均衡策略算法。
5. 配置：根据url上的配置规则生成配置信息
6. 分组聚合：合并返回结果。
7. 本地伪装：mork通常用于服务降级，mock只在出现非业务异常(比如超时，网络异常等)时执行

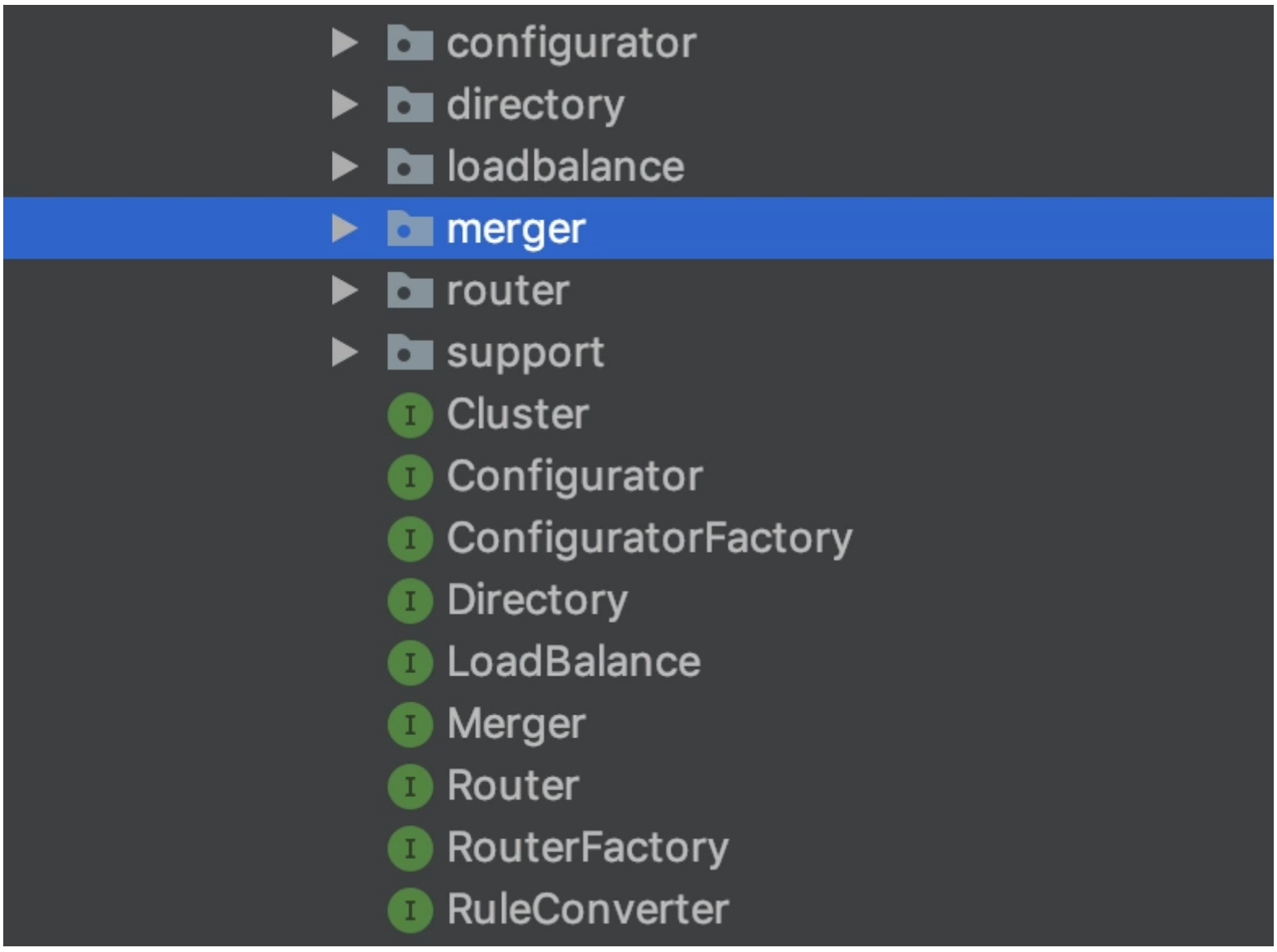
以上几部分跟《dubbo源码解析（一）Hello,Dubbo》的"（二）dubbo-cluster——集群模块"介绍有些类似，这里再重新讲一遍是为了明确我接下来介绍集群模块的文章内容分布，也就是除了本文之外，我会用七篇文章来讲解以上的七部分，不管内容多少，只是为了把相应内容区分开来，能让读者有选择性的阅读。

在官方网站上有一段介绍我觉得写的非常的好：

集群工作过程可分为两个阶段，第一个阶段是在服务消费者初始化期间，集群 Cluster 实现类为服务消费者创建 Cluster Invoker 实例，即上图中的 merge 操作。第二个阶段是在服务消费者进行远程调用时。以 FailoverClusterInvoker 为例，该类型 Cluster Invoker 首先会调用 Directory 的 list 方法列举 Invoker 列表（可将 Invoker 简单理解为服务提供者）。Directory 的用途是保存 Invoker，可简单类比为 List<invoker>。其实现类 RegistryDirectory 是一个动态服务目录，可感知注册中心配置的变化，它所持有的 Inovker 列表会随着注册中心内容的变化而变化。每次变化后，RegistryDirectory 会动态增删 Inovker，并调用 Router 的 route 方法进行路由，过滤掉不符合路由规则的 Invoker。当 FailoverClusterInvoker 拿到 Directory 返回的 Invoker 列表后，它会通过 LoadBalance 从 Invoker 列表中选择一个 Inovker。最后 FailoverClusterInvoker 会将参数传给 LoadBalance 选择出的 Invoker 实例的 invoker 方法，进行真正的远程调用。

本文要来讲的无非就是这几部分内容的一个大概，并且介绍一下这几部分内容涉及到的接口。集群的包结构我就不在这里展示了,就是《dubbo源码解析（一）Hello,Dubbo》的"（二）dubbo-cluster——集群模块"中的图片。下面我们直接对应各个部分来介绍相应的接口源码。

目录



关于目录介绍请查看《dubbo源码解析（一）Hello,Dubbo》的"（二）dubbo-cluster——集群模块"介绍。

源码分析

（一）Cluster

```
@SPI(FailoverCluster.NAME)
public interface Cluster {

    /**
     * Merge the directory invokers to a virtual invoker.
     * 将目录调用程序合并到虚拟调用程序。
     * @param <T>
     * @param directory
     * @return cluster invoker
     * @throws RpcException
     */
    @Adaptive
    <T> Invoker<T> join(Directory<T> directory) throws RpcException;

}
```

该接口是集群容错接口，可以看到它是一个可扩展接口，默认实现FailoverCluster，当然它还会有其他的实现，每一种实现都代表了一种集群容错的方式，具体有哪些，可以看下面文章的介绍，他们都在support包下面，在本文只是让读者知道接口的定义。那么它还定义了一个join方法，作用就是把Directory对象变成一个 Invoker 对象用来后续的一系列调用。该Invoker代表了一个集群实现。似懂非懂就够了，后面看具体的实现会比较清晰。

（二）Configurator

```
public interface Configurator extends Comparable<Configurator> {

    /**
     * get the configurator url.
     * 配置规则，生成url
     * @return configurator url.
     */
    URL getUrl();

    /**
     * Configure the provider url.
     * 把规则配置到URL中
     *
     * @param url - old rovider url.
     * @return new provider url.
     */
    URL configure(URL url);

}
```

该接口是配置规则的接口，定义了两个方法，第一个是配置规则，并且生成url，第二个是把配置配置到旧的url中，其实都是在url上应用规则。

（三）ConfiguratorFactory

```
@SPI
public interface ConfiguratorFactory {

    /**
     * get the configurator instance.
     * 获得configurator实例
     * @param url - configurator url.
     * @return configurator instance.
     */
    @Adaptive("protocol")
    Configurator getConfigurator(URL url);

}
```

该接口是Configurator的工厂接口，定义了一个getConfigurator方法来获得Configurator实例，比较好理解。

（四）Directory

```
public interface Directory<T> extends Node {

    /**
     * get service type.
     * 获得服务类型
     * @return service type.
     */
    Class<T> getInterface();

    /**
     * list invokers.
     * 获得所有服务Invoker集合
     * @return invokers
     */
    List<Invoker<T>> list(Invocation invocation) throws RpcException;

}
```

该接口是目录接口，Directory 代表了多个 Invoker，并且它的值会随着注册中心的服务变更推送而变化 。一个服务类型对应一个Directory。定义的两个方法也比较好理解。

（五）LoadBalance

```
@SPI(RandomLoadBalance.NAME)
public interface LoadBalance {

    /**
     * select one invoker in list.
     * 选择一个合适的调用，并且返回
     * @param invokers   invokers.
     * @param url        refer url
     * @param invocation invocation.
     * @return selected invoker.
     */
    @Adaptive("loadbalance")
    <T> Invoker<T> select(List<Invoker<T>> invokers, URL url, Invocation invocation) throws RpcException;

}
```

该接口是负载均衡的接口，dubbo也提供了四种负载均衡策略，也会在下面文章讲解。

（六）Merger

```
@SPI
public interface Merger<T> {

    /**
     * 合并T数组，返回合并后的T对象
     * @param items
     * @return
     */
    T merge(T... items);

}
```

该接口是分组聚合，将某对象数组合并为一个对象。

（七）Router

```
public interface Router extends Comparable<Router> {

    /**
     * get the router url.
     * 获得路由规则的url
     * @return url
     */
    URL getUrl();

    /**
     * route.
     * 筛选出跟规则匹配的Invoker集合
     * @param invokers
     * @param url        refer url
     * @param invocation
     * @return routed invokers
     * @throws RpcException
     */
    <T> List<Invoker<T>> route(List<Invoker<T>> invokers, URL url, Invocation invocation) throws RpcException;

}
```

该接口是路由规则的接口，定义的两个方法，第一个方法是获得路由规则的url，第二个方法是筛选出跟规则匹配的Invoker集合。

（八）RouterFactory

```
@SPI
public interface RouterFactory {

    /**
     * Create router.
     * 创建路由
     * @param url
     * @return router
     */
    @Adaptive("protocol")
    Router getRouter(URL url);

}
```

该接口是路由工厂接口，定义了获得路由实例的方法。

后记

> 该部分相关的源码解析地址：https://github.com/CrazyHZM/i...

该文章大致讲解了dubbo中集群模块的内容，并且讲解了相关接口的设计。接下来我将开始对cluster集群模块中的集群容错部分，也就是support中的源码进行讲解。

crazyhzm

265

关注作者

0 条评论

得票 • 时间

撰写评论 ...

提交评论

推荐阅读

dubbo注册服务IP解析异常及IP解析源码分析

在使用dubbo注册服务时会遇到IP解析错误导致无法正常访问.比如:本机设置的IP为172.16.11.111,但实际解析出来的是180.20.174.1...

平常 • 阅读 37

dubbo源码解析（一）Hello,Dubbo

你好，dubbo，初次见面，我想和你交个朋友。先给出一套官方的说法：ApacheDubbo是一款高性能、轻量级基于Java的RPC开源...

CrazyHzm • 阅读 633

聊聊Dubbo - Dubbo可扩展机制源码解析

摘要： 在Dubbo可扩展机制实战中，我们了解了Dubbo扩展机制的一些概念，初探了Dubbo中LoadBalance的实现，并自己实现了...

猫耳 • 阅读 21