dubbo 支持的9种协议

xiaojin21cen 2018-04-06 15:58:45 24875 收藏 40

分类专栏： dubbo

版权

文章目录

一、9种协议

1、dubbo 协议 (默认)

2、rmi 协议

3、hessian 协议

4、http 协议

5、webservice 协议

6、thrift 协议

7、memcached 协议

8、redis 协议

9、rest ( 就是 RestFull)

二、多协议配置

不同服务使用不同协议

同时使用多种协议

三、面试题：

Dubbo支持dubbo、rmi、hessian、http、webservice、thrift、redis等多种协议，但是Dubbo官网是推荐我们使用Dubbo协议的。

下面我们就针对Dubbo的每种协议详解讲解，以便我们在实际应用中能够正确取舍。

一、9种协议

1、dubbo 协议 (默认)

1、dubbo 缺省协议 采用单一长连接和NIO异步通讯，适合于小数据量大并发的服务调用，以及服务消费者机器数远大于服务提供者机器数的情况

2、不适合传送大数据量的服务，比如传文件，传视频等，除非请求量很低。

缺省协议，使用基于mina1.1.7+hessian3.2.1的tbremoting交互。

特性

连接个数：单连接

连接方式：长连接

传输协议：TCP

传输方式：NIO异步传输

序列化：Hessian 二进制序列化

适用范围：传入传出参数数据包较小（建议小于100K），消费者比提供者个数多，单一消费者无法压满提供者，尽量不要用dubbo协议传输大文件或超大字符串。

适用场景：常规远程服务方法调用

配置

<!--配置协议： -->

<dubbo:protocol name="dubbo" port="20880" />

<!--设置默认协议： -->

<dubbo:provider protocol="dubbo" />

<!-- 设置服务协议： -->

<dubbo:service protocol="dubbo" />

<!-- 多端口 -->

<dubbo:protocol id="dubbo1" name="dubbo" port="20880" />

<dubbo:protocol id="dubbo2" name="dubbo" port="20881" />

<!-- 配置协议选项： -->

<dubbo:protocol name=“dubbo” port=“9090” server=“netty” client=“netty” codec=“dubbo”

serialization=“hessian2” charset=“UTF-8” threadpool=“fixed” threads=“100”

queues=“0” iothreads=“9” buffer=“8192” accepts=“1000” payload=“8388608” />

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

3、Dubbo协议缺省每服务每提供者每消费者使用单一长连接，如果数据量较大，可以使用多个连接。

<dubbo:protocol name="dubbo" connections="2" />

1

<dubbo:service connections=”0”>或<dubbo:reference connections=”0”>表示该服务使用JVM共享长连接。(缺省)

<dubbo:service connections=”1”>或<dubbo:reference connections=”1”>表示该服务使用独立长连接。

<dubbo:service connections=”2”>或<dubbo:reference connections=”2”>表示该服务使用独立两条长连接。

4、为防止被大量连接撑挂，可在服务提供方限制大接收连接数，以实现服务提供方自我保护

<dubbo:protocol name="dubbo" accepts="1000" />

1

常见问题

为什么要消费者比提供者个数多？

因dubbo协议采用单一长连接，假设网络为千兆网卡(1024Mbit=128MByte)，根据测试经验数据每条连接最多只能压满7MByte(不同的环境可能不一样，供参考)，理论上1个服务提供者需要20个服务消费者才能压满网卡

为什么不能传大包？

因dubbo协议采用单一长连接，如果每次请求的数据包大小为500KByte，假设网络为千兆网卡(1024Mbit=128MByte)，每条连接最大7MByte(不同的环境可能不一样，供参考)，单个服务提供者的TPS(每秒处理事务数)最大为：128MByte / 500KByte = 262。单个消费者调用单个服务提供者的TPS(每秒处理事务数)最大为：7MByte / 500KByte = 14。如果能接受，可以考虑使用，否则网络将成为瓶颈。

为什么采用异步单一长连接？

因为服务的现状大都是服务提供者少，通常只有几台机器，而服务的消费者多，可能整个网站都在访问该服务，比如Morgan的提供者只有6台提供者，却有上百台消费者，每天有1.5亿次调用，如果采用常规的hessian服务，服务提供者很容易就被压跨，通过单一连接，保证单一消费者不会压死提供者，长连接，减少连接握手验证等，并使用异步IO，复用线程池，防止C10K问题。

接口增加方法，对客户端无影响，如果该方法不是客户端需要的，客户端不需要重新部署；

输入参数和结果集中增加属性，对客户端无影响，如果客户端并不需要新属性，不用重新

部署；

输入参数和结果集属性名变化，对客户端序列化无影响，但是如果客户端不重新部署，不管输入还是输出，属性名变化的属性值是获取不到的。

总结：服务器端 和 客户端 对 领域对象 并不需要完全一致，而是按照最大匹配原则。

如果不是集成Spring，单独配置如下：

dubbo.service.protocol=dubbo

1

2、rmi 协议

1、RMI协议采用JDK标准的java.rmi.\*实现，采用阻塞式短连接和JDK标准序列化方式 。

注：

如果正在使用RMI提供服务给外部访问（公司内网环境应该不会有攻击风险），同时应用里依赖了老的common-collections包（dubbo不会依赖这个包，请排查自己的应用有没有使用）的情况下，存在反序列化安全风险。

请检查应用：

将commons-collections3 请升级到3.2.2版本：

https://commons.apache.org/proper/commons-collections/release\_3\_2\_2.html

将commons-collections4 请升级到4.1版本：

https://commons.apache.org/proper/commons-collections/release\_4\_1.html

新版本的commons-collections解决了该问题 。

特性

连接个数：多连接

连接方式：短连接

传输协议：TCP

传输方式：同步传输

序列化：Java标准二进制序列化

适用范围：传入传出参数数据包大小混合，消费者与提供者个数差不多，可传文件。

适用场景：常规远程服务方法调用，与原生RMI服务互操作

接口

如果服务接口继承了java.rmi.Remote接口，可以和原生RMI互操作，即：

提供者用Dubbo的RMI协议暴露服务，消费者直接用标准RMI接口调用，或者提供方用标准RMI暴露服务，消费方用Dubbo的RMI协议调用。

如果服务接口没有继承java.rmi.Remote接口，缺省Dubbo将自动生成一个com.xxx.XxxService$Remote的接口，并继承java.rmi.Remote接口，并以此接口暴露服务，

但如果设置了<dubbo:protocol name="rmi" codec="spring" />，将不生成$Remote接口，而使用Spring的RmiInvocationHandler接口暴露服务，和Spring兼容。

配置

<!-- 定义 RMI 协议： -->

<dubbo:protocol name="rmi" port="1099" />

<!--设置默认协议： -->

<dubbo:provider protocol="rmi" />

<!-- 设置服务协议： -->

<dubbo:service protocol="rmi" />

<!--多端口： -->

<dubbo:protocol id="rmi1" name="rmi" port="1099" />

<dubbo:protocol id="rmi2" name="rmi" port="2099" />

<dubbo:service protocol="rmi1" />

<!--Spring 兼容性：-->

<dubbo:protocol name="rmi" codec="spring" />

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

3、hessian 协议

Hessian 协议用于集成 Hessian 的服务，Hessian 底层采用 Http 通讯，采用 Servlet 暴露服务，Dubbo 缺省内嵌 Jetty 作为服务器实现。

Dubbo 的 Hessian 协议可以和原生 Hessian 服务互操作，即：

提供者用 Dubbo 的 Hessian 协议暴露服务，消费者直接用标准 Hessian 接口调用

或者提供方用标准 Hessian 暴露服务，消费方用 Dubbo 的 Hessian 协议调用。

Hessian 是 Caucho 开源的一个 RPC 框架，其通讯效率高于 WebService 和 Java 自带的序列化。

特性

连接个数：多连接

连接方式：短连接

传输协议：HTTP

传输方式：同步传输

序列化：Hessian二进制序列化

适用范围：传入传出参数数据包较大，提供者比消费者个数多，提供者压力较大，可传文件。

适用场景：页面传输，文件传输，或与原生hessian服务互操作

依赖:

<dependency>

<groupId>com.caucho</groupId>

<artifactId>hessian</artifactId>

<version>4.0.7</version>

</dependency>

1

2

3

4

5

约束

1、参数及返回值需实现Serializable接口

2、参数及返回值不能自定义实现List, Map, Number, Date, Calendar等接口，只能用JDK自带的实现，因为hessian会做特殊处理，自定义实现类中的属性值都会丢失。

配置

<!-- 定义 hessian 协议： -->

<dubbo:protocol name="hessian" port="8080" server="jetty" />

<!--设置默认协议： -->

<dubbo:provider protocol="hessian" />

<!-- 设置 service 协议： -->

<dubbo:service protocol="hessian" />

<!-- 多端口：-->

<dubbo:protocol id="hessian1" name="hessian" port="8080" />

<dubbo:protocol id="hessian2" name="hessian" port="8081" />

<!--直连：-->

<dubbo:reference id="helloService" interface="HelloWorld" url="hessian://10.20.153.10:8080/helloWorld" />

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

web.xml 配置:

<servlet>

<servlet-name>dubbo</servlet-name>

<servlet-class>com.alibaba.dubbo.remoting.http.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dubbo</servlet-name>

<url-pattern>/\*</url-pattern>

</servlet-mapping>

1

2

3

4

5

6

7

8

9

注意：如果使用servlet派发请求

协议的端口<dubbo:protocol port="8080" />必须与servlet容器的端口相同，

协议的上下文路径<dubbo:protocol contextpath="foo" />必须与servlet应用的上下文路径相同。

4、http 协议

基于http表单的远程调用协议。参见：[HTTP协议使用说明]

特性

连接个数：多连接

连接方式：短连接

传输协议：HTTP

传输方式：同步传输

序列化：表单序列化 ，即 json

适用范围：传入传出参数数据包大小混合，提供者比消费者个数多，可用浏览器查看，可用表单或URL传入参数，暂不支持传文件。

适用场景：需同时给应用程序和浏览器JS使用的服务。

配置

<!-- 配置协议：-->

<dubbo:protocol name="http" port="8080" />

<!-- 配置 Jetty Server (默认)：-->

<dubbo:protocol ... server="jetty" />

<!-- 配置 Servlet Bridge Server (推荐使用)： -->

<dubbo:protocol ... server="servlet" />

1

2

3

4

5

6

7

8

配置 DispatcherServlet：

<servlet>

<servlet-name>dubbo</servlet-name>

<servlet-class>org.apache.dubbo.remoting.http.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dubbo</servlet-name>

<url-pattern>/\*</url-pattern>

</servlet-mapping>

1

2

3

4

5

6

7

8

9

注意:如果使用 servlet 派发请求 ：

协议的端口<dubbo:protocol port="8080" />必须与servlet容器的端口相同，

协议的上下文路径<dubbo:protocol contextpath="foo" />必须与servlet应用的上下文路径相同。

5、webservice 协议

基于 WebService 的远程调用协议，基于 Apache CXF的 frontend-simple 和 transports-http 实现。

可以和原生 WebService 服务互操作，即：

提供者用 Dubbo 的 WebService 协议暴露服务，消费者直接用标准 WebService 接口调用，

或者提供方用标准 WebService 暴露服务，消费方用 Dubbo 的 WebService 协议调用。

依赖

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-frontend-simple</artifactId>

<version>2.6.1</version>

</dependency>

<dependency>

<groupId>org.apache.cxf</groupId>

<artifactId>cxf-rt-transports-http</artifactId>

<version>2.6.1</version>

</dependency>

1

2

3

4

5

6

7

8

9

10

特性

连接个数：多连接

连接方式：短连接

传输协议：HTTP

传输方式：同步传输

序列化：SOAP文本序列化

适用场景：系统集成，跨语言调用

1、基于CXF的 frontend-simple 和 transports-http 实现。

2、CXF是Apache开源的一个RPC框架：http://cxf.apache.org，由Xfire和Celtix合并而来 。

可以和原生WebService服务互操作，即：

提供者用Dubbo的WebService协议暴露服务，消费者直接用标准WebService接口调用，或者提供方用标准WebService暴露服务，消费方用Dubbo的WebService协议调用。

约束

参数及返回值需实现Serializable接口

参数尽量使用基本类型和POJO。

配置

<!-- 配置协议： -->

<dubbo:protocol name="webservice" port="8080" server="jetty" />

<!-- 配置默认协议：-->

<dubbo:provider protocol="webservice" />

<!-- 配置服务协议：-->

<dubbo:service protocol="webservice" />

<!--多端口： -->

<dubbo:protocol id="webservice1" name="webservice" port="8080" />

<dubbo:protocol id="webservice2" name="webservice" port="8081" />

<!-- 直连： -->

<dubbo:reference id="helloService" interface="HelloWorld" url="webservice://10.20.153.10:8080/com.foo.HelloWorld" />

<!-- WSDL -->

http://10.20.153.10:8080/com.foo.HelloWorld?wsdl

<!-- Jetty Server: (默认) -->

<dubbo:protocol ... server="jetty" />

<!-- Servlet Bridge Server: (推荐) -->

<dubbo:protocol ... server="servlet" />

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

配置 DispatcherServlet：

<servlet>

<servlet-name>dubbo</servlet-name>

<servlet-class>com.alibaba.dubbo.remoting.http.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dubbo</servlet-name>

<url-pattern>/\*</url-pattern>

</servlet-mapping>

1

2

3

4

5

6

7

8

9

注意:如果使用servlet派发请求：

协议的端口<dubbo:protocol port=“8080” />必须与servlet容器的端口相同，

协议的上下文路径<dubbo:protocol contextpath=“foo” />必须与servlet应用的上下文路径相同。

6、thrift 协议

当前 dubbo 支持的 thrift 协议是对 thrift 原生协议 [2] 的扩展，在原生协议的基础上添加了一些额外的头信息，比如 service name，magic number 等。

使用 dubbo thrift 协议同样需要使用 thrift 的 idl compiler 编译生成相应的 java 代码，后续版本中会在这方面做一些增强。

依赖

<dependency>

<groupId>org.apache.thrift</groupId>

<artifactId>libthrift</artifactId>

<version>0.8.0</version>

</dependency>

1

2

3

4

5

配置

<dubbo:protocol name="thrift" port="3030" />

1

常见问题

Thrift不支持null值，不能在协议中传null

7、memcached 协议

基于 memcached实现的 RPC 协议。

注册 memcached 服务的地址

RegistryFactory registryFactory = ExtensionLoader.getExtensionLoader(RegistryFactory.class).getAdaptiveExtension();

Registry registry = registryFactory.getRegistry(URL.valueOf("zookeeper://10.20.153.10:2181"));

registry.register(URL.valueOf("memcached://10.20.153.11/com.foo.BarService?category=providers&dynamic=false&application=foo&group=member&loadbalance=consistenthash"));

1

2

3

在客户端引用

在客户端使用 ：

<dubbo:reference id="cache"

interface="http://10.20.160.198/wiki/display/dubbo/java.util.Map" group="member" />

1

2

或者点对点直连：

<dubbo:reference id="cache"

interface="http://10.20.160.198/wiki/display/dubbo/java.util.Map"

url="memcached://10.20.153.10:11211" />

1

2

3

自定义接口：

<dubbo:reference id="cache"

interface="com.foo.CacheService" url="memcached://10.20.153.10:11211" />

1

2

方法名建议和memcached的标准方法名相同，即：get(key), set(key, value), delete(key)。

如果方法名和memcached的标准方法名不相同，则需要配置映射关系：(其中”p:xxx”为spring的标准p标签)

<dubbo:reference id="cache" interface="com.foo.CacheService"

url="memcached://10.20.153.10:11211" p:set="putFoo"

p:get="getFoo" p:delete="removeFoo" />

1

2

3

8、redis 协议

基于 Redis实现的 RPC 协议。

注册 redis 服务的地址

可以通过脚本或监控中心手工填写表单注册redis服务的地址：

RegistryFactory registryFactory = ExtensionLoader.getExtensionLoader(RegistryFactory.class).getAdaptiveExtension();

Registry registry = registryFactory.getRegistry(URL.valueOf("zookeeper://10.20.153.10:2181"));

registry.register(URL.valueOf("redis://10.20.153.11/com.foo.BarService?category=providers&dynamic=false&application=foo&group=member&loadbalance=consistenthash"));

1

2

3

在客户端引用

在客户端使用：

<dubbo:reference id="store"

interface="http://10.20.160.198/wiki/display/dubbo/java.util.Map" group="member" />

1

2

或者，点对点直连：

<dubbo:reference id="store" interface="http://10.20.160.198/wiki/display/dubbo/java.util.Map"

url="redis://10.20.153.10:6379" />

1

2

也可以使用自定义接口：

<dubbo:reference id="store" interface="com.foo.StoreService" url="redis://10.20.153.10:6379" />

1

方法名建议和redis的标准方法名相同，即：get(key), set(key, value), delete(key)。

如果方法名和redis的标准方法名不相同，则需要配置映射关系：(其中”p:xxx”为spring的标准p标签)

<dubbo:reference id="cache" interface="com.foo.CacheService"

url="memcached://10.20.153.10:11211"

p:set="putFoo" p:get="getFoo" p:delete="removeFoo" />

1

2

3

9、rest ( 就是 RestFull)

基于标准的Java REST API——JAX-RS 2.0（Java API for RESTful Web Services的简写）实现的REST调用支持

参考： http://dubbo.apache.org/zh-cn/docs/user/references/protocol/rest.html

二、多协议配置

Dubbo 允许配置多协议，在不同服务上支持不同协议或者同一服务上同时支持多种协议。

不同服务使用不同协议

不同服务在性能上适用不同协议进行传输，比如大数据用短连接协议，小数据大并发用长连接协议

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">

<dubbo:application name="world" />

<dubbo:registry id="registry" address="10.20.141.150:9090" username="admin" password="hello1234" />

<!-- 多协议配置 -->

<dubbo:protocol name="dubbo" port="20880" />

<dubbo:protocol name="rmi" port="1099" />

<!-- 指定使用 dubbo协议 -->

<dubbo:service interface="com.alibaba.hello.api.HelloService" version="1.0.0" ref="helloService" protocol="dubbo" />

<!-- 指定使用 rmi协议 -->

<dubbo:service interface="com.alibaba.hello.api.DemoService" version="1.0.0" ref="demoService" protocol="rmi" />

</beans>

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

同时使用多种协议

需要与 http 客户端互操作

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd">

<dubbo:application name="world" />

<dubbo:registry id="registry" address="10.20.141.150:9090" username="admin" password="hello1234" />

<!-- 多协议配置 -->

<dubbo:protocol name="dubbo" port="20880" />

<dubbo:protocol name="hessian" port="8080" />

<!-- 使用多个协议暴露服务 -->

<dubbo:service id="helloService" interface="com.alibaba.hello.api.HelloService" version="1.0.0" protocol="dubbo,hessian" />

</beans>

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

三、面试题：

1、dubbo推荐用什么协议？

默认使用 dubbo 协议。

2、dubbo 默认使用什么序列化框架，你知道的还有哪些？

dubbo 有多种协议，不同的协议默认使用不同的序列化框架。比如：

dubbo 协议 默认使用 Hessian2 序列化。（说明：Hessian2 是阿里在 Hessian 基础上进行的二次开发，起名为Hessian2 ）

rmi协议 默认为 java 原生序列化，

http 协议 默认为 为 json 。

此外补充，

hessian 协议，默认是 hessian 序列化；

webservice 协议，默认是 soap 文本序列化 。

上面说的是 协议默认使用的序列化，也可以指定使用第三方的序列化框架，如 Kryo 、 FST等序列化框架，详情参考：

https://blog.csdn.net/xiaojin21cen/article/details/90269659

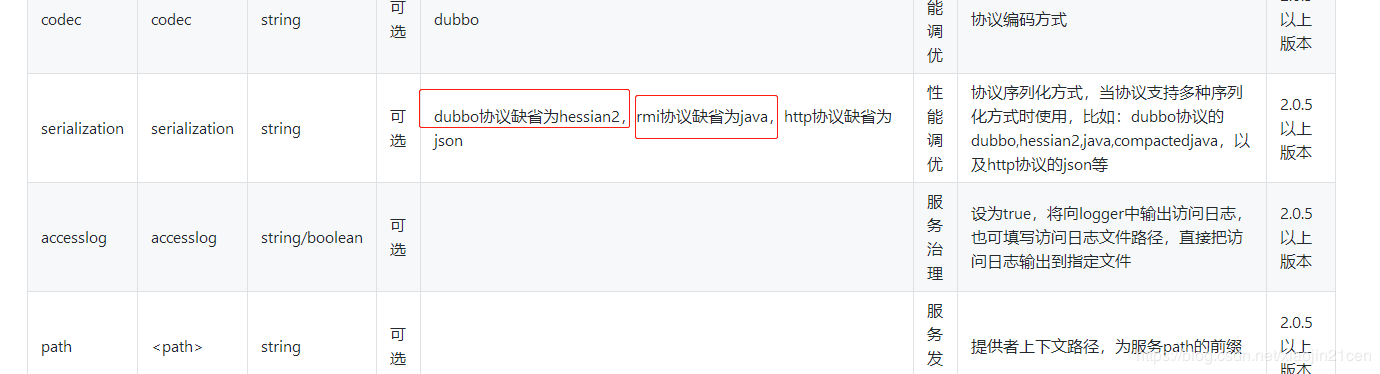
https://blog.csdn.net/xiaojin21cen/article/details/90269952

知识来源：

上面的回答在dubbo 官方文档中有说明 。 http://dubbo.apache.org/zh-cn/docs/user/references/xml/dubbo-protocol.html



往下滚动：



如上图所示，在 serialization 这一行中，dubbo协议默认是hessian2 序列化框架 ，

rmi协议默认为 java原生序列化 ；http 协议默认为 json 序列化。

————————————————

版权声明：本文为CSDN博主「xiaojin21cen」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/xiaojin21cen/java/article/details/79834222