文章编号：2095-2163(2019)02-0182-05 中图分类号：TP311.52 文献标志码：A

Dubbo 的序列化协议扩展及其 RPC 协议 Thrift 的优化

翟成彤，董海峰
（西安石油大学 计算机学院，西安 710065）

摘 要：基于开源的分布式服务架构 Dubbo，这篇论文扩展了高效的二进制序列化和反序列化协议 protostuff，同时提高了传输的效率并且优化了 RPC 的传输调用协议 Thrift，以提高异构系统之间 RPC 通信性能和实现。

关键词：Dubbo；Thrift 协议；Protostuff 协议；序列化

Dubbo´s serialization protocol extension and its protocol Thrift optimization of RPC

ZHAI Chengtong，DONG Haifeng
（School of Computer Science，Xi´an Shiyou University，Xi´an 710065，China）

【Abstract】 Based on the open source distributed service architecture Dubbo，this paper extends the serialization protocol Protostuff，an efficient binary serialization and deserialization protocol，improves the efficiency of transmission，and optimizes the transmission call protocol Thrift of RPC to improve RPC communication performance and implementation between heterogeneous systems.

【Key words】 Dubbo；Thrift protocol；Protostuff protocol；serialization

1 概述

Dubbo 是一个专注于提供高性能和透明 RPC 远程服务调用的解决方案，同时也是阿里巴巴内部 SOA 服务治理的核心框架，每天为 2000+服务提供超过 30 亿次调用并且常常广泛见于移动互联网+的应用中[1]。

随着技术日新月异的发展，出现了许多在序列化和跨平台服务调用方面优秀的新技术。如果能够把这些优秀的技术整合到 Dubbo 架构中，那么就可以提升 Dubbo 的序列化性能和跨平台调用的能力。

在远程调用方面，使用更高效的序列化协议 Protostuff 编码和解码，以提高传输数据的性能和减少系统调用响应的时间。此外，在互联网公司可能会使用不同的语言去开发软件，同时为了提高代码的复用性和异构系统之间的互相调用，开源社区团队基于 Dubbo 扩展了 REST 协议，用来在异构系统之间进行通讯。但是，该类模式却是通过（HTTP+JSON/XML）文本的方式传输数据，性能并不是很好[2]。如果使用跨语言的 Thrift 二进制协议，可以极大地提高在异构系统之间调用的性能。

2 背景

2.1 Protostuff and Hessian2 对比

Dubbo 默认的序列化协议 Hessian2 相比基于 Protobuf 的 Protostuff 协议更慢[3]，因此，Protostuff 协议可以被添加到 Dubbo 的默认 RPC 序列化协议栈中，用来更加高效地提升 Dubbo RPC 传输性能。

2.2 Dubbo 中的 Thrift 协议和原生的 Thrift 协议比较

Dubbo 协议栈默认有一种 Thrift 协议的实现，但却不是原生的 Thrift 协议，因为有一个协议头被添加到了 Thrift 协议，破坏了原生 Thrift 协议，使得解析协议比原生协议更为繁复与耗时。如果实践中需要考虑用到跨语言通信，最好的方式是使用原生的 Thrift 协议，因其具有更好的跨平台特性。故而，本文拟尝试去重构 Dubbo RPC 协议栈中的 Thrift 协议以便支持原生的 Thrift 协议，给框架带来更高的性能和跨平台特性。

2.3 Dubbo 系统调用基本模型

Dubbo 系统间调用如图 1 所示。由图 1 可知，该模型设计中的重点功能阐释详见如下。

（1）服务提供者在启动时，即在注册中心注册。

（2）服务消费者启动，则向注册中心订阅服务。

（3）注册中心为消费者返回服务提供者列表，如果提供者发生了改变，注册中心就会基于长连接再将改变的数据推送到消费者。

（4）服务消费者从提供者列表中，使用负载均衡算法选择一个提供者加以调用，如果调用失败，然后选择另外一个。

作者简介：翟成彤（1991-），男，硕士研究生，主要研究方向：管理信息系统；董海峰（1965-），男，博士，副教授，主要研究方向：信息管理系统。

收稿日期：2018-10-29

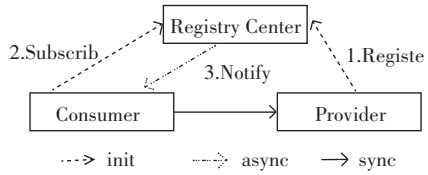

图 1　Dubbo 系统间调用

Fig. 1　Dubbo call between systems

3　扩展设计与实现

Dubbo 使用微内核和插件化的结构，这样的设计表现出扩展性强的特点。Dubbo 的可扩展性原则是基于 JDK 的 SPI(service provider interface)[4]，SPI 是通过自定义@ SPI 注解和扩展点加载类 ExtensionLoader 来实现的。

ExtensionLoader 通过解析指定目录/META－INF/services/下的接口扩展文件加载被标记为@ SPI 接口的实现类。

因此，有 2 个步骤来扩展注释@ SPI 注解的接口。分别是：实现接口；在工程目录/META－INF/services/path 下，创建一个文件名为接口全路径名的文本文件。

首先，在 github 上下载 dubbox 源代码，然后在本地 idea 编辑器打开由 maven 构建的工程结构。本节主要研究了 dubbo－common 模块和 dubbo－rpc 模块的基础。在所有扩展被实现后，整个 dubbox 项目编译为 dubbo.jar 包。对此可做探讨分述如下。

3.1　扩展 protostuff 序列化协议

3.1.1　Protosutff 协议实现扩展

研究可知，在 dubbo－common 模块中进行扩展包支持的序列化，就需要建立一个包来扩展 protostuff 协议，在这个包中需要实现 Dubbo 架构的 3 个核心接口的序列化，即：com.alibaba.dubbo.serialize.Serialization；com. alibaba. dubbo. common. serialize. ObjectInput；com. alibaba.dubbo.common.serialize.ObjectOutput；扩展序列化协议名称定义为 protostuff，并支持 protostuff1.0.8。在这个包中，设有 5 个类，也就是：ProtostuffObjectInput、ProtostuffObjectOutput、ProtostuffSerialization、Serialization Util、ProtostuffFactory。其中 ProtostuffObjectInput 实现 ObjectInput 接口，处理输入流的反序列化；ProtostuffObjectOutput 实现 ObjectOuput 接口，处理输出流的序列化；ProtostuffSerialization 实现 Serialization 接口，定义序列化名称、序列化编号，并且实现序列化和反序列化函数。而 SerializationUtil，ProtostuffFactory 这 2 个类则是设计调用 protostuff 1.0.8.jar来处理函数的序列化和反序列化。接口和实现类的关系如图 2 所示。序列化数据的实现过程如图 3 所示。反序列化数据实现过程如图 4 所示。

在此基础上，项目的根目录/resours/META－INF/dubbo/ com. alibaba. dubbo. common. serialize. Serialization 就需要添加一行，内容表述如下：fileprotostuff = com. alibaba. dubbo. common. serialize. support.protostuff. ProtostuffSerialization

当获取到序列化接口的实现时，通过 SPI 机制，可以加载扩展的实现类。

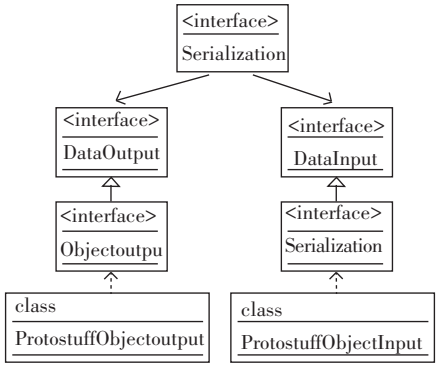

图 2　序列化接口和实现类关系

Fig. 2　Serialize interfaces and relationships of implementation class

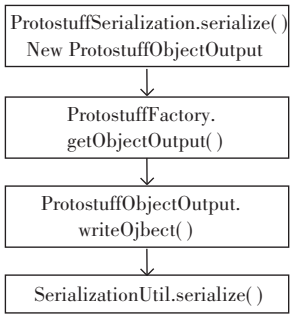

图 3　输出流数据序列化类的实现

Fig. 3　The implementation of output stream data serialization class

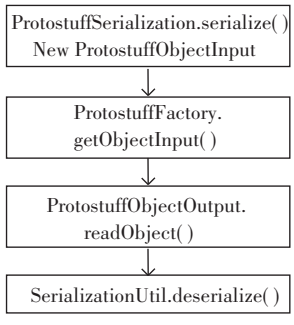

图 4　输入流反序列化类的实现

Fig. 4　The implementation of input stream deserialization class

3.1.2　Protostuff 配置和使用

Dubbo 在传输层实现数据的序列化和反序列化，也就是编码和解码。根据 Dubbo 的数据包结构特征，可以知道，当数据流到达传输层时可以读取在服务提供者的目录下 xml 形式的 protostuff 序列化协议配置，例如：<dubbo:protocol name="dubbo" Serialization="protostuff"/>，然后就会加载序列化实现 ProtostuffSerialization。当使用 ProtostuffSerialization 序列化和反序列化数据时，服务提供者将会对数据进行编码和解码。

服务提供者编码数据并提供给消费者的核心处理过程如图 5 所示。服务的提供者从服务消费者接

收到请求数据并解码的过程如图 6 所示。

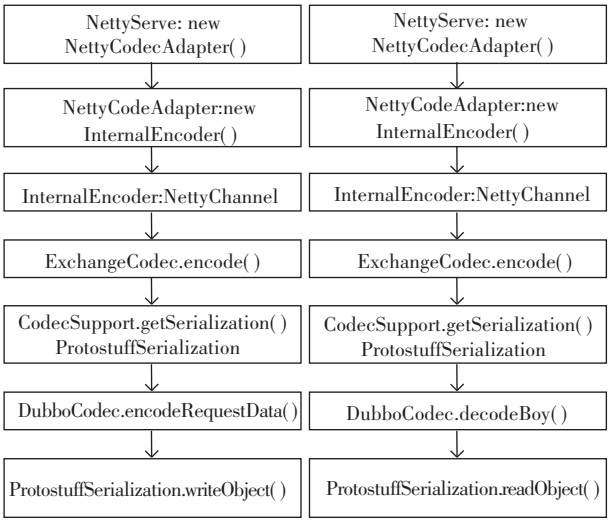

图 5 数据编码过程 **图 6 数据解码过程**

Fig. 5 Data encoding process **Fig. 6 Data decoding process**

3.2 RPC 的 Thrift 协议扩展

3.2.1 优化 Thrift 协议的实现

研究中,需要在 dubbox 项目目录下创建名字为 dubbo-rpc-thriftx 的模块,这个名字为了和以前 dubbo- rpc-thrift 作区分同时可以被兼容,符合设计模式开闭原则[5]。这里使用原生 thrift0.10.0.jar 作为支持,在此基础上原生协议可以使用,命名为 thriftX。在新近启用的模块目录下创建新的目录 com / alibaba / dubbo / rpc /protocol / thriftx,用来存储协议扩展类的实现,其中之一需要实现 RPC 通讯协议的核心—com.alibaba.dubbo.rpc. Protocol。在这个 thriftx 包中还需要实现 thriftXprotocol 和 ThriftUtils 这 2 个类。其中,ThriftUtils 是用来生产网络通讯所需对象的工程,thriftXprotocol 是 Protocol 接口的实现,可用于暴露远程服务和引用远程服务,具体的设计细节是 ThriftUtils 工具类调用 thrift1.10.jar 包中的方法实现。

研究可得,通过实现 thriftXprotocol 类发布服务的过程,如图 7 所示。通过实现 thriftXprotocol 类引用服务的过程,如图 8 所示。

接下来,在 src/main/resources/META-INF/dubbo/ internal/com.alibaba.dubbo.rpc.Protocol 路径下文本文件中加入 ThriftX = com.alibaba.dubbo.rpc.protocol.thriftx. ThriftXProtocol。当获得 RPC 通信协议接口的实现时,可以通过 SPI 机制将其加载到扩展实现类中。

3.2.2 ThriftX 协议的配置和使用

首先,定义一个 service.thrift 文件,再编译生成接口文件。然后拷贝文件到服务提供和服务消费者工程。

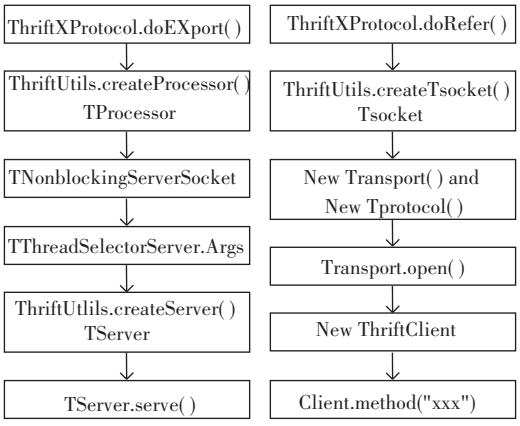

图 7 ThriftX 协议暴露服务过程 **图 8 ThriftX 协议引用服务的过程**

Fig. 7 ThritfX protocol exposure service process **Fig. 8 ThriftX protocol referencing service process**

在服务提供者中,接口的方法需要被实现并且接口需要被暴露,服务暴露的过程可分述如下。

(1)在服务启动时,spring 容器根据配置的 xml 文件[6],可将 <dubbo: servier protocol = " thriftX" interface = " com. alibaba. dubbo. demo. DemoService $ Iface"/>,解析到 ServiceConfig,而后通过 ServiceConfig 获取到格式如下格式的 URL: Registry://registry-addr/com. alibaba. dubbo. registry. RegistryService? Export = URL. encode (" thriftX://service-addr/com. alibaba. dubbo. demo. DemoService $ Iface? Version = 1.0.0")。

(2)通过 URL 中的"registry://"约定协议头的识别,则将会调用 RegistryProtocol 类中的 export( ) 方法,接着将读取 URL 中的参数注册到注册中心。服务提供者的 URL 格式如:" thriftX://service-addr/com. alibaba. dubbo. demo. DemoService $ Iface? Version = 1.0.0"。通过调用 URL 中的"thrift://"协议头,会调用 export( ) 方法打开服务端口。thriftX 暴露服务的过程如图 7 所示。

进一步研究可知,客户端服务调用过程可以分为 2 步,对此可阐述如下。

(1)客户端服务使用 dubbo 的方式调用 XML 文件中定义的服务引用的接口,服务引用过程的研究内容如下。

① 在 XML 配置文件中配置<dubbo:reference id = " DemoService" interface = " com. alibaba. dubbo. demo. DemoService $ Iface"/> ,根据 Spring 约束,XML 标签会被 ReferenceConfig 对象解析过后,通过此对象获取 url。

② 通过 URL"registry://"协议头识别，服务将会调用 RegistryProtocol 类中的 refer()方法，使得后续通过配置的参数查询提供者 url，例如：thriftX：//service-addr/com.alibaba.dubbo.demo.DemoService$Iface?Version=1.0.0.。在解析 thrifx://时，thriftXProtocol 中的 doRefer()方法会被调用，如图 8 所示。

(2)客户端可能是一种没有使用 dubbo 的状态，更甚至不是由 Java 语言编写的，这就需要根据特定的语言接口文件的需要，客户端使用不同定义的接口 thrift 文件。通过接口文件可以得到一个 Client 对象，客户端对象包含 stub 的所有接口函数，而用户代码可以通过 Client 对象来调用 thrift 文件的接口函数，事实上，Client 调用的是接口函数的本地 stub 接口。接口函数的 stub 将调用请求发送到服务提供者，同时服务提供者根据调用的函数名和函数参数调用实际的实现函数来进行特定的操作。Thrift 服务提供者在指令处理流程后，会将相应函数的返回值发送到调用客户端对象。Thrift 的客户端对象将函数的返回值传递给用户的调用函数以完成调用过程。

4　性能测试

4.1　测试环境

本文测试在同一台机器部署了 zookeeper，也就是服务提供者和服务消费者的笔记本电脑上。这台机器的配置是，Thinkpad-E450，memory 4 G，CPU intel i5 win7 操作系统和 jdk1.7.0_07。

4.2　测试数据(单线程)

(1)定义包含通用数据类型的 Stu 对象，例如：

```
public class Stu implements Serializable {
public String name;
public Integer age;
public double account;
public Date birth;
public Boolean high;
public long height;
//get set method
}
```

(2)测试数据类型

简单的对象 Stu

复杂对象(list<stu> 包含 100 个 stu 对象)

1 K 字符串

10 K 字符串

1 M 字符串(代表较大数据)

(3)测试设计。在测试不同协议的性能时，研究中并未考虑服务器处理特定服务的耗时操作，因此用于测试的服务提供者处理逻辑旨在返回客户端请求的数据。另外，单个请求的响应时间相对较小，不容易衡量。因此，这个测试使用循环调用相同的接口，然后得到平均的响应时间。

4.3　测试结果及结果分析

Dubbo 不同的通信协议和序列化方案相结合的结果见表 1。而对应于表 1 中使用测试方案的数字参见表 2。表 2 中的结果是测试不同数据类型。所有测试结果时间单位为 ms。

表 1 可以被切分成 2 部分。第一部分是前五个数字 a、b、c、d、e。通信协议是 dubbo，但是序列化是不同的方案，与表 2 中的测试结果相比，可以从 5 种不同的序列化中分析得出利弊。另一部分是后面的 3 个字符 f，g，h 被设计为跨平台通信，通过 3 种协议比较表 2 中的测试结果，由此可以推得结论，在跨平台通信协议中是最好的。

表 1　传输协议和序列化模式

Tab. 1　Transport protocol and serialization mode

序号	传输协议和序列化模式
a	<dubbo:protocolname="dubbo" server="netty" port="30000" serialization="hessian2" />
b	<dubbo:protocolname="dubbo" server="netty" port="30000" serialization="java" />
c	<dubbo:protocolname="dubbo" server="netty" port="30000" serialization="kyro" />
d	<dubbo:protocolname="dubbo" server="netty" port="30000" serialization="fst" />
e	<dubbo:protocolname="dubbo" server="netty" port="30000" serialization="protostuff" />
f	<dubbo:protocolname="rest" port="30000" server="tomcat"/>
g	<dubbo:protocolname="thrift" port="30000"/>
h	<dubbo:protocolname="thrift2" port="30000"/>

在本文中，平均响应时间是大于线上时间的[7]。不同的测试环境会导致测试结果出现一定的偏差，但在相同的环境中，根据测试结果的相对数据变化得出相应的结论。

通过比较表 2 中的 5 组数据 a、b、c、d、e。与测试结果中的默认序列化协议 hessian2 相比，Dubbox 扩展序列化协议 kyro 和 fst 的性能未见显著提高。本次研究扩展的 protostuff 有突出的性能表现，针对

此点,重点做了一些测试,结果都相差无几,其运行机理原因还需要重新分析。比较表 2 中的 f,g,h 可知,f,g 明显具有更大的响应耗时。f 组使用的是 HTTP+JSON 的 REST 协议,过程中运用了文本序列化和应用层 HTTP 作为传输协议。理论上讲,f 组性能是最差的,和测试结果一致。g 组是 Dubbo 内置的 Thrift 协议,虽然修改了原生的 Thrift 协议,但其结果性能表现却并不理想。通过这 3 组测试结果,可以看到扩展的 ThriftX 在跨平台通信和平均响应时间方面更好,并且同时也显示了 Thrift 原始生态协议的优势。

表 2 不同类型数据测试结果

Tab. 2 Different types of data test results

序号	POJO(stu)	List<stur>	1 K(String)	10K(String)	1 M(String)
a	4.29	5.96	3.90	131.76	1 254.81
b	4.84	6.78	4.14	97.67	1 174.63
c	5.44	6.49	4.12	143.91	1 324.71
d	4.91	5.44	3.97	246.01	1 321.65
e	3.41	5.02	4.62	126.45	1 154.82
f	8.43	12.75	8.12	300.36	2 174.41
g	10.84	18.77	13.70	421.68	2 031.54
h	2.32	3.98	2.65	80.72	876.66

5 结束语

在本文中,dubbo 默认协议的分布式体系结构是可扩展的,添加了一个新的序列化协议。测试并考查了新的序列化协议 protostuff 及其原始序列化协议的运行性能。通过测试结果,将 protostuff 与其它序列化协议进行比较分析可知,其在小数据传输方面具有一定的优势。另外,对于跨平台通信优化扩展的 ThriftX 协议,通过与 dubbox 的 REST 协议和 dubbo 原始 Thrift 协议的比较,ThriftX 性能具有明显的优势。

但是,本次研究也仍然存在一定不足,测试结果在平均响应时间上一般来说偏大,究其原因可能在于测试环境,未来也需要有针对地加以改进解决。而考虑到本次测试中硬件资源有限,并未能真正涉足有关多线程测试的研究。而且除了扩展协议的稳定性外,在其它方面也还需要进行后续设计上的优化与完善。

参考文献

[1] Apache Software Foundation. 阿里巴巴 Dubbo 在线开发手册[EB/OL]. [2018].http://dubbo.apache.org/zh-cn/.

[2] 当当网. 开源中国 Dubbox 开源社区[EB/OL]. [2014-10-23]. https://www.oschina.net/p/dubbox.

[3] 序列化框架对比. http://x-rip.iteye.com/blog/1555293/.

[4] ECKEL B. JAVA 编程思想[M]. 4th ed. 北京:机械工业出版社, 2007.

[5] FREEMAN E, FREEMAN E, SIERRA K, et al.设计模式[M]. 北京:中国电力出版社,2007.

[6] Pivotal Software, Inc. Spring 约束[EB/OL]. [2019].http://spring.io.

[7] 博客园. 性能测试[EB/OL].[2019]. http://www.cnblogs.com/lengfo/p/4293399.html.

[8] Apache Software Foundation. thrift 官方文档[EB/OL]. [2017]. https://thrift.apache.org/docs/.

[9] GitHub. Inc. Github 开源程序 Dubbox[EB/OL]. [2019]. https://github.com/dangdangdotcom/dubbox.

[10] GitHub. Inc. 开源项目[EB/OL]. [2019].shttps://github.com/yjmyzz/dubbox.

(上接第 181 页)

3 结束语

本文以《Web 前端开发技术》课程为例,系统探索了该门课程网络教学平台的建设思路,并进行了实际应用研究,对于提高该门课程以及相关课程的教学质量提供了参考思路。网络教学平台的应用绝非一蹴而就的项目课题,本文论述的《Web 前端开发技术》课程的网络教学平台建设虽然已有部分工作交付使用,但仍存在诸多亟待完善之处。例如制作更丰富的教学视频,供学生自主学习;完善和更新实践作业题目,紧密联系实际等。

参考文献

[1] 周雨青,叶善专,朱明,等. 由欧美 MOOCs 风暴,探析中国大学物理精品资源共享课程的建设与发展[J]. 物理与工程,2014, 24(1): 9-14.

[2] 孟祥宇,全江涛,刘云飞. 运用网络课程平台进行开放式教学[J]. 新课程研究(高等教育),2013(2): 57-58.

[3] 王惠. 网络教学平台系统的设计与实现[D]. 北京:北京交通大学,2014.

[4] 辽东学院信息工程学院. 辽东学院网络教学平台-Web 前端开发技术[EB/OL]. http://mooc1. chaoxing. com/course/200964041.html.