文章编号:1671-4598(2017)12-0279-04 DOI:10.16526/j.cnki.11—4762/tp.2017.12.072 中图分类号:TP391 文献标识码:A

基于 Thrift 的 RPC 中间件在航天信息系统中的设计与实现

杨 帆,孔维萍,蒋晓肖,肖永利,虢 华
(上海航天控制技术研究所,上海 200233)

摘要:随着互联网技术的发展,大数据存储技术日趋成熟,针对系统应用实施中存在的跨平台和可扩展性问题,分析了在航天信息系统中远程服务调用的几种技术方法,提出了基于 Thrift 技术设的一种跨平台可扩展的远程调用中间件,利用 Thrift 框架的技术特点,有效提高了可扩展性和高效传输性能;实际应用表明,该技术方案在跨平台可扩展性有显著优势,大数据传输性能优越,较好地满足信息化项目的需求。

关键词:Thrift;远程过程调用;航天信息化

Design and Implement of RPC Middle Ware Based on Thrift Frame in Aerospace Information System

Yang Fan, Kong Weiping, Jiang Xiaoxiao, Xiao Yongli, Guo Hua
(Shanghai Aerospace Control Technology Institute, Shanghai 200233, China)

Abstract: With the rapid development of internet, big data storage technology is gradually maturing. This paper analyze and compare several popular remote service invocation mechanisms in aerospace information application, then proposed a remote service invocation middleware based on the Thrift framework, which has a better transmission performance and expandability. The application indicates that, based on these techniques, the middleware is effective and flexible to expand, satisfying the needs of the projects.

Keywords: Thrift; RPC; aerospace information

0 引言

国家十三五以来,互联网、移动互联网、物联网等信息化技术高速发展,带动了航天军工系统的信息化应用,多数型号设备实验测试软件也逐步从单机单软、专机专用的定制化研发模式向通用化、信息化的技术发展。刘杰等[1]介绍了航天信息化工程中云计算方面的应用情况,胡明明等[2]探讨了航天系统中信息化通用标准体系的构建,马昌超等[3]介绍了航天信息化的建设模式。本文调研了主流互联网产品的应用情况,根据阿里巴巴公司的实时数据传输平台[4],以及 Facebook 的开源日志收集系统[5]等的信息技术研究,结合航天信息化系统应用背景,提出了一种基于 Thrift 框架的跨平台、跨语言的远程过程调用协议(remote procedure call protocol, RPC)方案,并在自研的信息化项目能源系统(energy mangement system, EMS)、试验管理系统(test data management system, TDM)中成功应用,取得良好的效果。

1 基于 Thrift 框架的远程服务调用分析

1.1 远程服务调用应用分析

目前,远程服务调用的技术实现方式大致可分为三类[6]:第一类,采用分布式对象技术;第二类,基于过程的服务调用;第三类,通过 WebService 服务调用方式。

分布式对象技术是最早期提出的一种实现方式,主要技术有:RMI(remote method invocation,远程方法调用)/EJB(enterprise java bean, JavaEE 服务器端组件模型),COM(component object model,组件对象模型)/DCOM(distribute component object model,分布式组件对象模型)/COM+和 OMG(object management group,对象管理组织)提出的 CORBA(common object request broker architecture,公用对象请求代理体系结构)[7]。其中比较流行的技术是微软提出的 COM/DCOM 和 OMG 对象管理组织提出的 CORBA。COM/DCOM/COM+是基于组件的对象模型,DCOM 比 COM 技术支持更多的通信协议,而且在分布式处理性能优于 COM,COM+则在延续 COM 技术的基础上增加了分布式网络应用程序的设计和实现[8]。CPRBA 技术广泛应用在分布式产品中,解决了分布式环境下软硬件系统的互连问题。实际应用中,上述技术均要求客户端和服务端应用明确的同类型对象协议模型,在跨语言和可扩展性方面的应用有较高难度。

基于过程的远程服务调用主要指 RPC,是基于客户端/服务器的分布式架构模型[9]。类似于本地 API,部署在客户端的应用程序(Client)通过 RPC 的服务请求调用部署在在网络中某节点的 RPC 服务器端(Server)的服务。目前,主流的 RPC 技术有 Facebook 在 2008 年开源的支持跨语言特性的服务部署框架 Apache Thrift,Hadoop 创始人 Doug Cutting 领导开发的数据序列化系统 Apache Avro 以及 Google 发布的自动序列化、反序列化结构化数据的机制 Protocol Buffers[10]。它们提供跨语言的序列化和通信支持,不仅解决了分布式对象技术的兼容性问题,同时在高并发,跨平台语言及大数据传输上有明显优势。

收稿日期:2017-04-12; **修回日期**:2017-05-10。

作者简介:杨 帆(1988-),女,硕士研究生,主要从事卫星地面测试系统以及信息化软件,大数据存储方向的研究。

Web Service 服务调用方式是通过 Web 访问实现，具有模块自包含和自描述特性的新型 Web 应用程序分支[11]。这种方式使用标准互联网协议，既是一种编程组件，又是一种远程服务调用方式。其中，HTTP 协议为最常用的传输协议，最常见的消息交换方式为 JSON－RPC、XML－RPC 以及 SOAP (Simple Object Access Protocol) 等[12]。而 Web Service 在传输过程中携带的冗余信息多，在编码和解码 xml 文件上处理耗时也较高，因此，在处理信息系统的并发应用中对服务器的负载较高，导致性能较低[13]。

1.2 Thrift 框架原理特点

经调研分析，Apache Thrift、Apache Avro 以及 Protocol Buffers 这 3 种主流的新型 RPC 框架在大数据传输、大规模并发调用中性能优异，尤其适合应用于云计算。考虑信息化系统中的对可扩展性，高并发和数据体量等方面的要求，进一步分析比较 3 种 RPC 技术方式的特点，得出 Thrift 框架最适合应用在信息化系统这种大型的的数据通信中间件搭建中，本文提出采用基于 Thrift 的 RPC 框架进行信息化系统中数据通信的中间件的设计方案。Thrift 的框架结构如图 1 所示。

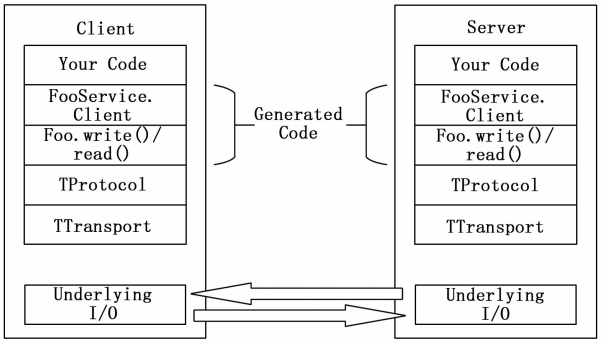

图 1 Thrift 框架技术结构示意图

如上图所示，Thrift 框架主要由 4 个部分组成，从上至下分别是应用层、框架层、协议层以及传输层。用户在应用层对业务逻辑进行自定义。Thrift 的 IDL 对用户定义的应用层数据结构进行编译，生成 Server/Client 端代码框架和读写操作，即图中 FooService. Client 和 Foo. Write () /Read ()。TProtocol 是负责管理不同类型数据编码方案以及序列化和反序列化的协议。TTransport 用于实现数据传输的传输层，底层 I/O 通信负责实际的数据传输。

基于 Thrift 框架的通信协议，是通过内部序列化机制对数据进行简化，将数据以对象形式进行传输，本地 Client 端通过过程声明来调用远程 Server 端的方式实现数据的交换，提高了传输效率，降低了通信成本。Thrift 结合其功能强大的软件堆栈和代码生成引擎，可以构建在 C＃，C＋＋，Java，Python，PHP，Ruby，JavaScript，Perl 等编程语言间实现高效无缝的服务[14]。Thrift 软件框架支持的数据类型如表 1 所示。

service 对应服务的类 Thrift 协议层通过数据的序列化和反序列化机制，如 JSON，XML，Plain Text，Binary，Compact Binary 等，抽象数据结构的定义。传输协议主要分为文本和二进制两大类，为提高传输效率，大多数传输协议采用二进制类型，常用的 TProtocol 通信协议如表 2 所示。

常用 Thrift 框架传输层如表 3 所示几种。

常见的服务端类型如表 4 所示。

表 1 数据类型表

数据类型	定义	描述
基本类型	bool	布尔值，true 或 false
	byte	8 位有符号整数
	i16	16 位有符号整数
	i32	32 位有符号整数
	double	64 位浮点数
	string	未知编码文本或二进制字符串
结构体类型	struct	定义公共对象，类似 C 语言中间结构体，Java 中 JavaBean
容器类型	list	对应 Java 的 ArrayList
	set	对应 Java 的 HashSet
	map	对应 Java 的 HashMap
异常类型	exception	对应异常
服务类型	service	对应服务的类

表 2 通信协议类型表

协议类型	描述
TBinaryProtocol	二进制编码传输协议
TCompactProtocol	采用 Zigzag 高效率的、密集的二进制编码格式，节省空间，传输效率更高
TJSONProtocol	JSON 的数据编码协议
TSimpleJSONProtocol	只提供 JSON 只写的协议，适用于通过脚本语言解析

表 3 传输层定义表

传输层定义	描述
TSocket	使用阻塞式 I/O 进行传输
TFramedTransport	使用非阻塞方式，按块的大小进行传输
TNonblockingTransport	使用非阻塞方式，用于构建异步客户端

表 4 服务端类型表

服务端定义	描述
TSimpleServer	单线程服务器端使用标准的阻塞式 I/O
TThreadPoolServer	多线程服务器端使用标准的阻塞式 I/O
TNonblockingTransport	多线程服务器端使用非阻塞式 I/O

综合分析比较，以 Thrift 框架作为信息系统通信中间件设计架构的主要技术解决方案，主要有以下优势特点：

1）跨平台语言，可扩展性强。Thrift 框架支持包括主流的 C＋＋ 、Java 、C＃ 、Python 、PHP 、Ruby 、JavaScript 及 Coaca 、Smalltalk、OCaml 等十余种编程语言，有效实现基于不同开发语言的应用系统的 Server 端和 Client 端无缝连接和数据交互，大大提高信息化系统的可扩展性。

2）高效性：相较于传统的 XML 和 JSON 格式和一般二进制数据编码格式而言，Thrift 具有更高效数据传输方式，支持更多类型对象进行序列化和反序列化，耗能更少。

3）易用性：Thrift 框架更容易被研发人员理解并应用。用户使用 Thrift 框架时，定义好描述数据传输协议、数据类型和服务的接口文件后，使用 Thrift 提供的编译器即可自动生成远程过程调用中间件的 Server 和 Client 端实现，协议层和传输

层代码也自动生成。因此，用户无需手动编写基于不同语言、不同系统结构的通信中间件代码文件，只需维护统一的接口文件即可。

2　两种 Thrift 的 RPC 中间件实现

基于 Thrift 框架的数据调用中间件实现主要分三步。首先要定义数据传输对象的标准格式，并定义接收或者发送数据的接口。Thrift 框架支持多种类型数据传输，可以是原始数据字节类型数组，基于 MySQL 的查询计算结构，也可以是文件数据流。第二步，通过 Thrift 代码生成引擎为各系统生成数据接口的底层实现。第三步，实现具体产品测试数据传输的业务功能。数据标准格式和服务接口格式是通过 Thrift 文件来描述的，该文件定义了数据的各项信息以及接口函数的原型。Thrift 根据该文件会生成特定版本的数据接口实现，开发者只需关心数据传输的业务。Thrift 引擎也可以编译出不同的语言版本，从而实现跨语言跨平台的调用。本节分别以 Java 和 C＃编程语言为版本介绍基于 Thrift 框架中间件部署方法实现。

2.1　基于 Java 的实现

本节以 Java 语言为例，介绍 Thrift 软件框架应用实现。实现中间件 Client 端和 Server 端部署时，需要在开发环境中引入 libthrift.jar，log4j.jar 等公共 jar 包，并编写创建 Thrift 的 IDL 文档 Test. thrift，执行 thrift－0.9.3.exe －gen java Test. thrift 命令，调用 Thrift 编译器生成 Test. java 公共文件，部署过程示意如图 2 所示。

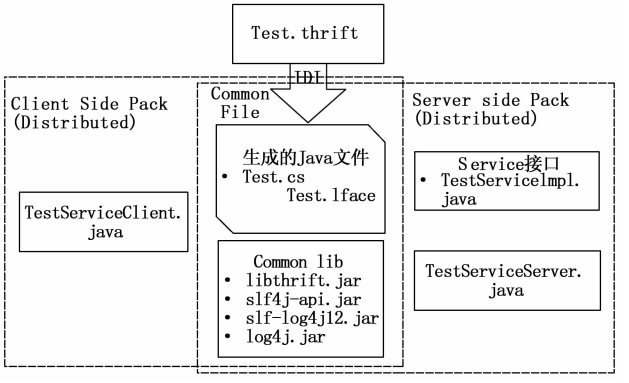

图 2　基于 Java 的 Thrift 框架部署图

如图 2 所示，Client 端调用 Server 端的 TestServiceClient.java 代码。Server 端的 Process 类对象实现 RPC 请求处理的核心 Test.Iface 接口，并启动 TestServiceServer.java。Server 端和 Client 端利用 Test.java 提供的 API 进行远程服务调用。

Thrift 服务器端的流程如下：

1）创建服务端口通道（TServerTransport）；

2）确定传输协议（TProtocolFactory）；

3）创建数据处理器（Processor）；

4）创建数据传输协议（Protocol）；

5）基于 Processor、TProtocolFactory 和 TServerTransport 创建服务器模型（Server）；

6）运行 Server。

此处框架的服务端采用高效密集的二进制编码格式 TCompactProtocol 进行数据传输，Server 端代码示例如下：

```
public static void main(String[] args) {
    try {
     TServerTransport serverTransport = new TServerSocket(7911);
    TProtocolFactory proFactory = new TCompactProtocol.Factory();
    TMultiplexedProcessor processor = new TMultiplexedProcessor();
    processor.registerProcessor("Test",new Test.Processor<Test.Iface>(new Test.Impl()));
    TServer server = new TThreadPoolServer(new Args(serverTransport).protocolFactory(proFactory).processor(processor));
    System.out.println("Start server on port 7911...");
    server.serve();
    } catch (Exception e) {
    /*异常处理*/
    }
    }
```

当服务需要处理大量数据更新时，Server 端使用 TThreadPoolServer 线程池服务模型，采用标准的阻塞式 I/O，预先创建一组线程处理请求，当有客户端连接过来时，从线程池里分配可用的连接处理客户端请求。

客户端通过调用服务端的方法传输定义好的 Test 对象。具体的步骤为：

1）创建数据传输方式（Transport）；

2）创建数据传输协议（Protocol）；

3）创建客户模型（Client）；

4）调用 Client 的相应方法。核心代码如下：

```
public static void main(String[] args) {
    try {
    TTransport transport=new TSocket(ip, port);
    Transport.open();
TProtocol protocol = new TMultiplexedProtocol(new TCompactPortocol(transport),"Test");
    ProductTestInformationService.Client client = new Test.Client(protocol);
/* 服务端提供的接口方法*/
transport.close();
    server.serve();
    } catch (TException e) {
    /*异常处理*/
    }
    }
```

2.2　基于 C＃的实现

基于 C＃语言实现信息系统中间件搭建时，需要在工程中引用 Thrift 开源框架编译生成的 Thrift.dll，根据用户定义的 Thrift 的 IDL 文档 Test.thrift，执行 thrift－0.9.3.exe －gen csharp Test.thrift 命令，调用 Thrift 编译器生成 Test.cs 公共文件，部署过程示意如图 3 所示。

Client 端调用 Server 端的 TestServiceClient.cs 代码。Server 端的实现 Test.Iface 接口，并启动 TestServiceServer.cs 服务。Server 端和 Client 端利用 Test.cs 提供的 API 进行远程服务调用。由于 C＃语言的限制无法使用非阻塞的多线程服务端，此处使用 TThreadPoolServer 线程池服务模型，采用标准的阻塞式 I/O。工程中 Server 端核心代码如下：

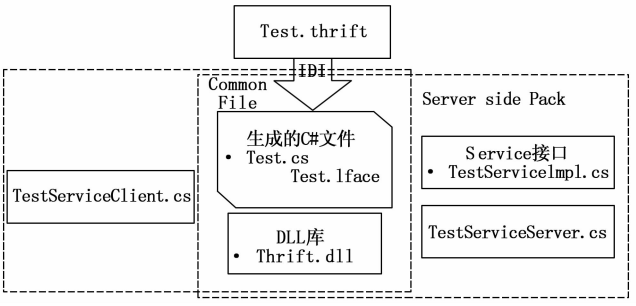

图 3 基于 C# 的 Thrift 框架部署图

```
public void Start()
{
try
{
TServerSocket serverTransport = new TServerSocket(9090, 0, false);
//一个端口注册多个服务
TMultiplexedProcessor processors = new TMultiplexedProcessor();
Test.Processor testProcessor = new Test.Processor(new Test Impl());
processors.RegisterProcessor("Test", testProcessor);
//多线程服务器
TServer server = new TThreadPoolServer(processors, serverTransport);
server.Serve();
}
catch (Exception x)
{
/*异常处理*/
}
}
```

Server 端使用 TThreadPoolServer 线程池服务模型，创建一组线程处理请求，当有 Client 端连接请求时，为其分配可用的连接处理线程。

客户端通过调用服务端的方法传输定义好的 Test 对象。具体的步骤为：

1）创建数据传输方式（Transport）；

2）创建数据传输协议（Protocol）；

3）创建客户模型（Client）；

客户端核心代码如下：

```
static void Main(string[] args)
{
try
{
TTransport transport = new TSocket("localhost", 9090);
TProtocol protocol = new TBinaryProtocol(transport);
Test.Client testClient = new Client.Client(protocol);
transport.Open();
Console.WriteLine("Test Client Calls");
transport.Close();
}
catch (Exception x)
{
/*异常处理*/
}
}
```

3 应用实例

近年来信息化技术迅速发展，航天应用系统紧跟时代步伐，所内自主研发了能源管理系统（energy mangement system，简称 EMS）、试验管理系统（test data management system，简称 TDM）等多个信息化项目。针对信息化平台特点，本文基于 Thrift 框架设计了灵活的跨语言数据通信远程服务调用中间件方案，有效解决信息化项目中各个系统技术架构迥异，协议开发难度大等问题。本文以 EMS 系统为例，将此方案在各信息化系统中进行广泛推广和应用。

能源管理系统是响应国家“十三五”号召，全方面为实现“指标能分解、责任能落实、监管能有效、考核有实效”目标，践行人人参与的节能减排，以信息化技术，科学实现能耗减排而自主研制的信息化系统。能源管理系统为航天研发，产品生产的节能用能管控提供良好的信息化管控手段，改变既有的人工干预管理的方式，采用硬件与软件结合，通过实时高效的采集电、水等计量仪表设备的数据，进行数据整理和统计分析，通过数据建模和图表绘制，以信息化手段进行数据管理，为能源的监控和管理提供强有力的数据支撑，及监管条件，有效实现精细化管理，提高工作效率。系统提供数据预警及设备报警功能，能随时发现跑、冒、滴、漏，有效地保证工业生产中的能源的高效利用，杜绝能源浪费。EMS 系统组成结构如图 4 所示。

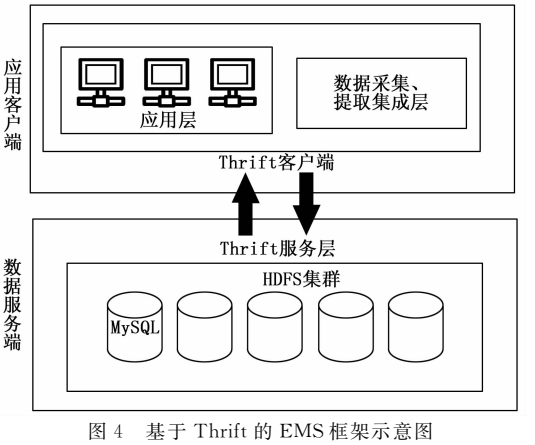

图 4 基于 Thrift 的 EMS 框架示意图

能源管理系统中数据存储层是采用 C# 语言实现，前台 B/S 端采用 Java 语言实现，本文提出的基于 Thrift 框架设计的远程调用中间件，有效的解决此类异构系统的接口服务调用问题，设计人员只需定义好数据传输对象格式，无需针对编程语言的差异额外开发协议转换模块，大大降低系统开发时间，提高开发效率。

能源管理系统中使用 Thrift 框架技术作为 B/S 应用层、数据采集提取层与数据存储中心的服务调用中间件，根据系统数据通信协议定义的对象服务 IDL 文件，通过 Thrift 提供的编译器生成接口代码。当前台用户在界面上通过 Client 端发起服务请求时，服务注册组件寻找业务的绑定信息并获取指定链接，调用 Server 端的服务，继而 Server 端返回调用结果，展

（下转第 306 页）

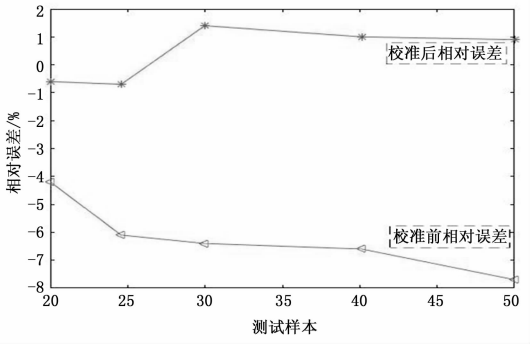

图 7 96%RH 时 5 组不同温度点测试样本相对误差

从表 3 可以看出，同一标准湿度 96%RH 下，未进行温度补偿时传感器的输出最大相对误差为－7.7%，国家湿度检定规程要求±7%RH（标准湿度 40%RH 以下或 70%RH 以上），经过本文的方法补偿修正后最大相对误差为 1.2%，图 7 是同一标准湿度点不同温度下补偿后的相对误差，与补偿修正前相比，传感器的输出准确度有很大的提升。

综合（1）和（2）两种情况，根据补偿前后的传感器输出特性曲线进行比较如图 6 和图 7，可以看出，经过补偿后精度远高于补偿前，实验仿真结果表明，传感器经过基于 L－M 算法的 BP 神经网络补偿后，输出更接近湿度标准值，说明该方法较准确的实现了湿度传感器的非线性补偿校正，达到了预期目标。

4 结束语

本文提出了湿度传感器补偿模型，针对其本身的输出非线性特性和由于温度的影响导致的非线性输出特性而引入补偿环节，利用 L－M 算法、拟牛顿算法、共轭梯度算法等 3 种算法的 BP 神经网络经行仿真验证，结果表明以 L－M 算法为基础的 BP 神经网络对湿度传感器输出进行补偿时效果最为优异，补偿校正后误差不大于 1.4%，湿度传感器的准确度有了显著的提高，满足了实际使用的需要，本文基于神经网络的传感器非线性补偿为传感器的校准提供了新的设计思路。

参考文献：

[1] 王 悦，叶海明. LabWindows/CVI 下基于 BP 神经网络的温度补偿虚拟湿度测量系统设计 [J]. 国外电子测量技术，2010，29 (1)：36－38.

[2] Wang X D，Ye M Y. Hysteresis and nonlinearity compensation of relative humidity sensor using support vector machines [J]. Sensors and ActuatorsB：chemical，2008，129 (1) 274－284.

[3] 俞阿龙. 基于 RBF 神经网络的热敏电阻温度传感器非线性补偿方法 [J]. 仪器仪表学报，2007，28 (5)：899－903.

[4] 孙文良，沈秋宇. HMP45D 温湿度传感器的检定校准 [J]. 气象水文海洋仪器，2009 (3)：124－126.

[5] 周胜海. 传感器非线性的硬件校正方法 [J]. 仪表技术与传感器，2002，21 (5)：1－4.

[6] 陈俊杰，卢 俊，黄唯一. 基于遗传神经网络的传感器系统的非线性校正 [J]. 仪器仪表学报，2003，24 (2)：201－204.

[7] 司端锋，常炳国，刘君华. 基于 BP 神经网络的传感器特性补偿新算法的研究 [J]. 仪表技术与传感器，2000 (1)：11－14.

[8] 顾 磊，黄庆安，秦 明. 一种新型 CMOS 兼容湿度传感器 [J]. 半导体学报，2004，25 (2)：174－178.

[9] 卢智远，周永军. 传感器非线性误差校正的 BP 神经网络方法研究 [J]. 传感器技术，2005，24 (2)：11－12.

[10] 刘 涛，王 华. 传感器非线性校正的遗传支持向量机方法 [J]. 电子测量与仪器学报，2011，25 (1)：56－60.

(上接第 282 页)

现给用户。Thrift 技术是整个业务调用的关键桥梁，为远程服务调用提供可靠性能保障。

4 结束语

本文通过分析常用的几种远程服务调用方法，提出基于 Thrift 实现的 RPC 中间件方案，并应用在航天信息化系统建设中，并以 C# 和 Java 两种语言构建典型的 Server/Client 应用，展示了 Thrift 框架的使用。信息化系统架构中数据通信中间件通过 Thrift 框架的跨平台跨语言以及数据高效传输等特点，提高信息化系统的可扩展性，大大降低系统的研发成本。相较于基于传统的 WebService 的解决方案，实施难度更低，研发成本更小，且具有更高的传输效率，尤其在大数据交换和存储方面表现优异，在信息化系统的实施中具有良好的应用前景。

参考文献：

[1] 刘 杰，丁向峰，柴旭东，等. 云计算在航天信息化工程中的应用与探讨 [J]. 军民两用技术与产品，2011 (10)：12－14.

[2] 胡明明，杨前进，段 晓. 航天器信息化标准体系构建初探 [J]. 航天标准化，2015 (1)：18－22.

[3] 马昌超，陈金兰. 航天信息化建设模式探讨 [A]. 中国宇航学会计算机应用专业委员会 2004 年学术交流会，2004.

[4] 周 豪. 大数据量下的实时数据报表系统的设计与实现 [D]. 北京：北京交通大学，2016.

[5] 李 俊. 基于块聚集的 MapReduce 性能研究与优化 [D]. 北京：北京交通大学，2014.

[6] 李 洋. 云计算中可扩展的远程服务调用机制的设计与实现 [D]. 哈尔滨：哈尔滨工业大学，2012.

[7] 胡 珉，许占文，张 宇. 用 Java RMI 实现 JDBC 远程调用的介绍 [J]. 沈阳工业大学学报，2003，25 (1)：65－68.

[8] David S. Platt. Understanding COM＋ [M]：55－79.

[9] 孙统风，孟现飞，姜利群. 基于 RPC 的 Agent 通信模型及其实现 [J]. 现代计算机，2003 (7)：10－13.

[10] 史栋杰. 5 种快速序列化框架的性能比较 [J]. 电脑知识与技术，2010 (34)：9710－971.

[11] 程晓飞. 基于 REST 架构的 Web Services 的研究与设计 [D]. 武汉：武汉理工大学，2010 (5)：15－23.

[12] 许卓明，栗 明，董逸生. 基于 RPC 和基于 REST 的 Web 服务交互模型比较分析 [J]. 计算机工程，1003，29 (20)：6－8.

[13] 袁 赟. Java 与 Restful Web Service. 电脑知识与技术（学术交流）[J]. 2007，4 (21)：780－782.

[14] 周 康，李 觊，董科军，等. 一种基于 Thrift 的日志收集分析系统 [J]. 科研信息化技术与应用，2015 (2).