(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 107301098 A
(43)申请公布日 2017.10.27

(21)申请号 201710451530.1

(22)申请日 2017.06.15

(71)申请人 搜易贷(北京)金融信息服务有限公司
地址 100080 北京市海淀区丹棱街1号院1号楼8层801室

(72)发明人 汪浩淼

(74)专利代理机构 北京市兰台律师事务所 11354
代理人 白伟 贾楠

(51)Int.Cl.
G06F 9/54(2006.01)
H04L 29/06(2006.01)
H04L 29/08(2006.01)

权利要求书2页 说明书6页 附图3页

(54)发明名称

一种基于Thrift协议的远程过程调用装置、方法及系统

(57)摘要

本发明属于通信及互联网金融领域,特别涉及一种基于Thrift协议的远程过程调用装置、方法及系统。所述装置包括API模块、远程调用模块;其中API模块包括若干预设服务约定接口及若干Thrift调用基础类;所述远程调用模块包括特定开发语言对应的若干基础工具类,可读取API模块中包含的预设服务约定接口,并结合API包中Thrift编译器生成的基础类,生成一个代理对象,并扩展spring将生成的代理对象注入到spring所管理的context中;调用者使用时,可通过spring注解机制获取到代理对象,并通过代理对象以thrift协议触发对远程服务的调用。调用者调用手写代码书写量大幅减少,降低业务组之间服务相互调用的开发门槛与成本。

CN 107301098 A

1.一种基于Thrift协议的远程过程调用装置，包括API模块、远程调用模块(TFCilentFactory)；其中API模块包括若干预设服务约定接口及若干Thrift调用基础类，一个预设服务约定接口对应一种远程服务，所述远程调用模块(TFCilentFactory)包括若干Java基础工具类，所述Java基础工具类可读取API模块中包含的预设服务约定接口，所述API模块中各预设服务约定接口通过url和tprotocol两个变量与一种特定远程服务对应，url用于定义该特定远程服务的地址，tprotocol用于定义该特定远程服务所以使用的Thrift网络数据传输格式，所述远程调用模块(TFCilentFactory)中的Java基础类工具可利用java反射机制读取任何API包中约定格式的接口，并提取出该接口对应的服务的url和网络传输格式tprotocol，结合API包中thrift编译器生成的基础代码，利用java的动态代理技术，根据约定接口动态生成一个代理对象，Java基础类工具包含经扩展的spring BeanFactoryPostProcessor后置处理器，该处理器中设有自定义处理逻辑，用以将代理对象注入到spring所管理的context中供调用者通过spring注解机制获取代理对象，通过代理对象以thrift协议触发对远程服务的调用。

2.根据权利要求1所述的基于Thrift协议的远程过程调用装置，其特征在于：还包括服务端模块和客户端模块，所述服务端模块包括基于thrift协议的若干预设服务，所述服务端模块可根据客户端模块发出的调用请求向调用者反馈对应的预设服务处理结果，并记录执行数据，所述执行数据包括输入流、输出流、基于thrift协议的预设服务、调用成功情况、响应时间中的一种或多种。

3.一种基于Thrift协议的远程过程调用方法，其特征在于，包括以下步骤：

步骤1：建立Thrift中间描述文件，其中定义了若干远程服务；

步骤2：定义一个注解，注解里面定义两个变量，其中一个变量用于定义各服务的远程地址(url)，另一个变量用于定义远程服务所使用的Thrift网络数据传输格式(tprotocol)；

步骤3：将步骤2中产生的注解存放于一个公共的基础包中；

步骤4：定义一个约定接口，并继承由Thrift编译器根据步骤1中的IDL文件所生成的对应服务的接口，并在约定接口上打上注解同时定义好步骤2中的两个变量；

步骤5：利用Thrift编译器和thrift-maven-plugin插件，将步骤4中的接口与对应的thrift中间描述文件所生成的thrift调用基础类工具打包成API包；

步骤6：建立远程调用包(TFCilentFactory)，所述远程调用包为Java开发语言对应的若干基础类工具，该等工具类可利用java反射原理读取API包中任一约定格式的接口，并提取出该接口对应服务的注解变量，结合API包中thrift编译器生成的基础代码，利用java的动态代理技术，根据约定接口动态生成一个代理对象，供业务开发者实现对特定服务的本地化远程调用，具体为：扩展Java基础类工具中的spring BeanFactoryPostProcessor后置处理器，添加自定义处理逻辑使引入了API包和远程调用包的工程可以在spring容器启动后自动扫描所有该工程引入的约定接口，并利用扫描到的约定接口生成对应的代理对象，将产生的代理对象自动存入spring context；业务开发者在需要使用该服务的类中就可以创建约定接口类型的成员变量，并在其上打@Autowired注解，从而自动注入代理对象，实现对特定服务的本地化远程调用。

4.根据权利要求3所述的基于Thrift协议的远程过程调用方法，其特征在于：还包括步

骤7,远程过程调用方法被执行时利用Hystrix进行包装,收集对应服务的统计数据,统计数据至少包括成功次数、失败次数、失败比例、调用时长中的一种或多种。

5.根据权利要求4所述的基于Thrift协议的远程过程调用方法,其特征在于:统计数据以一定频率将统计数据发往时序存储数据库influxDB,通过grafana以图表形式实时显示相关数据。

6.根据权利要求3-5中任一项所述的基于Thrift协议的远程过程调用方法,其特征在于,还包括步骤8,定义一个服务端处理工具包,主要入口需提供三个参数:输入流、输出流、thrift服务具体实现的实例,处理逻辑读取API包产生的通信格式tprotocol,并结合thrift编译器生成的基础类以及thrift服务具体实现实例,将输入流读取为请求对象,调用具体实现实例进行处理,处理结果回写入输出流,完成一次调用。

7.根据权利6所述的基于Thrift协议的远程过程调用方法,其特征在于:还包括步骤9,具体为扩展默认的远程服务所使用的Thrift网络数据传输格式变量,实现一个自定义的远程服务所使用的Thrift网络数据传输格式变量处理器,在扩展的远程服务的变量中再解析传输协议时增加详细的日志监控,包括服务端的调用性能,传递的参数数据。

8.一种基于Thrift协议的远程过程调用系统,其特征在于,包括服务端和调用端;所述服务端包括权利要求1-2中任一项所述基于Thrift协议的远程过程调用装置,且设有基于thrift协议的若干预设服务,所述服务端可根据调用端发送的请求,向调用者反馈对应的预设服务;所述调用端,可供调用者根据所述基于Thrift协议的远程过程调用装置生成的代理对象,向服务端发送调用请求,并接受服务端反馈的预设服务;所述服务端在处理调用请求的同时记录执行数据,所述执行数据包括输入流、输出流、基于thrift协议的预设服务、调用成功情况、响应时间中的一种或多种;所述系统采用HTTP通讯协议,并基于Nginx构架。

9.根据权利要求8所述的基于Thrift协议的远程过程调用系统,其特征在于,还至少包括以下装置中的一种或多种:Hystrix封装装置,时序存储数据库(influxDB),grafana装置,

其中:

Hystrix封装装置,用于在所述服务端执行调用请求时收集对应服务的统计数据,统计数据至少包括成功次数、失败次数、失败比例、调用时长中的一种或多种;

时序存储数据库(influxDB),用于按照一定频率接收和存储所述统计数据;

grafana装置,用于以图表形式实时显示时序存储数据库接收的统计数据。

10.根据权利要求8或9所述的基于Thrift协议的远程过程调用系统,其特征在于,还包括Thrift网络数据传输格式变量处理器,用于扩展默认的远程服务所使用的Thrift网络数据传输格式变量,通过扩展的远程服务的变量中在解析传输协议时增加详细的日志监控,包括服务端的调用性能,传递的参数数据。

一种基于Thrift协议的远程过程调用装置、方法及系统

技术领域

[0001] 本发明属于通信及互联网金融领域,特别涉及一种基于Thrift协议的远程过程调用装置、方法及系统。

背景技术

[0002] Thrift是由facebook开源出来用在系统以及各语言之间的进行RPC通信的基础软件框架,Thrift通过一个中间描述文件以及Thrift特有的中间描述语言来定义数据类型和服务接口,通过Thrift的编译器可以利用中间描述文件来生成各种语言的基础代码,从而实现各个语言以及系统间的无缝调用。Thrift编译器可在C++,Java,Python,PHP,Ruby,Erlang,Perl,Haskell,C#,Cocoa,JavaScript,Node.js,Smalltalk,OCaml,Delphi间提供支持。总得来说,Thrift提供了很完备的网络通信和中间协议。参考出处:http://thrift.apache.org/;http://dongxicheng.org/search-engine/thrift-framework-intro/。

[0003] 但是利用Thrift进行服务和客户端开发,需要开发人员懂得使用Thrift开发语言,对于不熟悉thrift环境的开发人员,存在较大的学习成本,不利于高效开发;此外,采用thrift原生开发方式,需要在开发环境中带入thrift代码,存在诸多不便。

发明内容

[0004] 为了克服现有技术中存在的如下技术缺陷:1、上手成本偏高,需要所有开发人员都要关心Thrift的客户端如何开发,无法高效的开发和使用;2、用原生的开发方式开发会让业务代码侵入很多关于Thrift本身的代码,使得业务代码非常臃肿、不清晰、不易维护以及影响代码可读性;3、缺乏灵活调度和监控调用过程方案;本发明在Thrift基础上继续封装,充分利用Thrift优秀的协议框架基础之上,开发一套可以和spring技术体系高度融合、让开发人员透明使用、几乎无学习成本、简单高效开发、无任何Thrift代码侵入、灵活调度、充分监控调用过程数据的远程过程调用系统。

[0005] 本发明首先提供了一种基于Thrift协议的远程过程调用装置,包括API模块、远程调用模块(TFCilentFactory);其中API模块包括若干预设服务约定接口及若干Thrift调用基础类,一个预设服务约定接口对应一种远程服务,所述远程调用模块(TFCilentFactory)包括若干Java基础工具类,所述Java基础工具类可读取API模块中包含的预设服务约定接口,并结合API包中的Thrift编译器生成的基础类,生成一个代理对象,并扩展Java基础类工具中的spring将生成的代理对象注入到spring所管理的context中,供调用者通过spring注解机制获取代理对象,通过代理对象以thrift协议触发对远程服务的调用。

[0006] 优选地,所述API模块中各预设服务约定接口通过url和tprotocol两个变量与一种特定远程服务对应,url用于定义该特定远程服务的地址,tprotocol用于定义该特定远程服务所以使用的Thrift网络数据传输格式,所述远程调用模块(TFCilentFactory)中的Java基础类工具可利用java反射机制读取任何API包中约定格式的接口,并提取出该接口

对应的服务的url和网络传输格式tprotocol，结合API包中thrift编译器生成的基础代码，利用java的动态代理技术，根据约定接口动态生成一个代理对象。根据项目涉及及目标调用环境编程语言需求，相关变量或参数的数量可以根据实际需要进行调整。

[0007] 优选地，Java基础类工具包含经扩展的springBeanFactoryPostProcessor后置处理器，该处理器中设有自定义处理逻辑，用以将代理对象注入到spring所管理的context中，供调用者通过spring注解机制获取代理对象，通过代理对象以thrift协议触发对远程服务的调用。

[0008] 优选地，还包括服务端模块和客户端模块，所述服务端模块包括基于thrift协议的若干预设服务，所述服务端模块可根据客户端模块发出的调用请求向调用者反馈对应的预设服务处理结果，并记录执行数据，所述执行数据包括输入流、输出流、基于thrift协议的预设服务、调用成功情况、响应时间中的一种或多种。

[0009] 本发明还提供了一种基于Thrift协议的远程过程调用方法，包括以下步骤：

[0010] 建立API包的步骤，所述API包中若干预设服务约定接口及Thrift调用基础类工具，一个预设服务约定接口对应一种远程服务；

[0011] 建立远程调用包(TFCilentFactory)的步骤，所述远程调用包设有特定开发语言对应的若干基础类工具，所述特定开发语言对应的基础类工具可读取所述API包中包含的预设服务约定接口，并协同Thrift调用基础类工具产生的基础代码，动态生成一个代理对象。

[0012] 本发明还提供了另一种基于Thrift协议的远程过程调用方法，包括以下步骤：

[0013] 步骤1:建立Thrift中间描述文件，其中定义了若干服务；

[0014] 步骤2:对中间文件定义一个注解，注解里面定义两个变量，其中一个变量用于定义各服务的远程地址(url)，另一个变量用于定义远程服务所使用的Thrift网络数据传输格式(tprotocol)；

[0015] 步骤3:将步骤2中产生的注解存放于一个公共的基础包中；

[0016] 步骤4:定义一个约定接口，并继承对应服务的接口，并在接口上打上注解定义好步骤2中的两个变量；

[0017] 步骤5:利用Thrift编译器和thrift-maven-plugin插件，将步骤4中的接口与对应的thrift中间描述文件所生成的thrift调用基础类工具打包成API包；

[0018] 步骤6:建立远程调用包(TFCilentFactory)，所述远程调用包设有特定开发语言对应的若干基础类工具，所述特定开发语言对应的基础类工具可读取API模块中包含的预设服务约定接口，并协同Thrift调用基础类工具产生的基础代码，动态生成一个代理对象，供业务开发者实现对特定服务的本地化远程调用。

[0019] 优选地，在前述基于Thrift协议的远程过程调用方法中，步骤6中的远程调用包为Java开发语言对应的若干基础类工具，该等工具类可利用java反射原理读取API包中任一约定格式的接口，并提取出该接口对应服务的注解变量，结合API包中thrift编译器生成的基础代码，利用java的动态代理技术，根据约定接口动态生成一个代理对象。

[0020] 优选地，在前述基于Thrift协议的远程过程调用方法中，扩展spring的BeanFactoryPostProcessor后置处理器，添加自定义处理逻辑，使引入了API包和远程调用包的工程可以在spring容器启动后自动扫描所有该工程引入的约定接口，并利用扫描到的

约定接口生成对应的代理对象，将产生的代理对象自动存入spring context；供业务开发者在需要使用该服务的类中就可以创建约定接口类型的成员变量，并在其上约定接口上打@Autowired，注解，即可快捷方便的自动注入代理对象，实现对特定服务的本地化远程调用。

[0021] 优选地，在前述基于Thrift协议的远程过程调用方法中，还包括步骤7，远程过程调用方法被执行时利用Hystrix进行包装，收集对应服务的统计数据，统计数据至少包括成功次数、失败次数、失败比例、调用时长中的一种或多种。

[0022] 优选地，在前述基于Thrift协议的远程过程调用方法中，统计数据以一定频率将统计数据发往时序存储数据库influxDB，通过grafana以图表形式实时将显示相关数据。

[0023] 优选地，在前述任一种基于Thrift协议的远程过程调用方法中，还包括步骤8，定义一个服务端处理工具包，主要入口需提供三个参数：输入流、输出流、thrift服务具体实现的实例，处理逻辑读取API包产生的通信格式tprotocol，并结合thrift编译器生成的基础类以及thrift服务具体实现实例，将输入流读取为请求对象，调用具体实现实例进行处理，处理结果回写入输出流，完成一次调用。

[0024] 优选地，在前述任一种基于Thrift协议的远程过程调用方法中，好包括步骤9，具体为扩展默认的远程服务所使用的Thrift网络数据传输格式变量，实现一个自定义的远程服务所使用的Thrift网络数据传输格式变量处理器，在扩展的远程服务的变量中再解析传输协议时增加详细的日志监控，包括服务端的调用性能，传递的参数数据。

[0025] 本发明还提供了一种基于Thrift协议的远程过程调用系统，包括服务端和调用端；所述服务端包括前述任一种基于Thrift协议的远程过程调用装置，且设有基于thrift协议的若干预设服务，所述服务端可根据调用端发送的请求，向调用者反馈对应的预设服务；所述调用端，可供调用者根据所述基于Thrift协议的远程过程调用装置生成的代理对象，向服务端发送调用请求，并接受服务端反馈的预设服务。

[0026] 优选地，在前述基于Thrift协议的远程过程调用系统中，所述服务端在处理调用请求的同时记录执行数据，所述执行数据包括输入流、输出流、基于thrift协议的预设服务、调用成功情况、响应时间中的一种或多种。

[0027] 优选地，在前述基于Thrift协议的远程过程调用系统中，所述系统采用HTTP通讯协议，并基于Nginx构架。

[0028] 优选地，在前述基于Thrift协议的远程过程调用系统中，还包括Hystrix封装装置，用于在所述服务端执行调用请求时收集对应服务的统计数据，统计数据至少包括成功次数、失败次数、失败比例、调用时长中的一种或多种。

[0029] 优选地，在前述基于Thrift协议的远程过程调用系统中，还包括时序存储数据库(influxDB)，用于按照一定频率接收和存储所述统计数据。

[0030] 优选地，在前述基于Thrift协议的远程过程调用系统中，还包括grafana装置，用于以图表形式实时显示时序存储数据库接收的统计数据。

[0031] 优选地，在前述任一种基于Thrift协议的远程过程调用系统中，还包括Thrift网络数据传输格式变量处理器，用于扩展默认的远程服务所使用的Thrift网络数据传输格式变量，通过扩展的远程服务的变量中再解析传输协议时增加详细的日志监控，包括服务端的调用性能，传递的参数数据。

[0032] 通过本发明所述技术方案，至少能够实现以下有益效果：

[0033] 1、客户端调用手写代码书写量以及服务端注入Thrift实现的代码量由平均5行降为1行，开发效率直接提升5倍左右；

[0034] 2、业务代码中没有任何Thrift相关代码的体现，实现了对Thrift调用实现代码的完全隔离，对业务开发者完全透明，大大降低了业务组之间服务相互调用的开发门槛与成本，并且完全不影响任何代码可读性；

[0035] 3、能实时观察各个接口调用数据，掌握各服务实时吞吐量和健康情况，方便制定扩容和维护等策略；

[0036] 4、采用http协议通讯，基于nginx可以方便快速的进行流量调度。

附图说明

[0037] 图1是本发明所述基于Thrift协议的远程过程调用装置结构示意图；

[0038] 图2是本发明所述基于Thrift协议的远程过程调用系统及方法实施例调用过程流程图；

[0039] 图3是本发明所述基于Thrift协议的远程过程调用系统网络结构图；

[0040] 图4是本发明所述基于Thrift协议的远程过程调用系统实施例效果示意图。

具体实施方式

[0041] 为了使本发明技术方案更容易理解，现结合附图采用具体实施例的方式，对本发明的技术方案进行清晰、完整的描述。应当注意，在此所述的实施例仅为本发明的部分实施例，而非本发明的全部实现方式，所述实施例只有示例性，其作用只在于为审查员及公众提供理解本发明内容更为直观明了的方式，而不是对本发明所述技术方案的限制。在不脱离本发明构思的前提下，所有本领域普通技术人员没有做出创造性劳动就能想到的其它实施方式，及其它对本发明技术方案的简单替换和各种变化，都属于本发明的保护范围。

[0042] 如图1所示，基于Thrift协议的远程过程调用装置，包括API模块、远程调用模块(TFCilentFactory)；其中API模块包括若干预设服务约定接口及若干Thrift调用基础类，一个预设服务约定接口对应一种远程服务，所述远程调用模块(TFCilentFactory)包括若干Java基础工具类，所述Java基础工具类可读取API模块中包含的预设服务约定接口，并结合API包中的Thrift编译器生成的基础类，生成一个代理对象，并扩展Java基础类工具中的spring将生成的代理对象注入到spring所管理的context中，供调用者通过spring注解机制获取代理对象，通过代理对象以thrift协议触发对远程服务的调用。

[0043] 所述API模块中各预设服务约定接口通过url和tprotocol两个变量与一种特定远程服务对应，url用于定义该特定远程服务的地址，tprotocol用于定义该特定远程服务所以使用的Thrift网络数据传输格式，所述远程调用模块(TFCilentFactory)中的Java基础类工具可利用java反射机制读取任何API包中约定格式的接口，并提取出该接口对应的服务的url和网络传输格式tprotocol，结合API包中thrift编译器生成的基础代码，利用java的动态代理技术，根据约定接口动态生成一个代理对象。根据项目涉及及目标调用环境编程语言需求，相关变量或参数的数量可以根据实际需要进行调整。扩展spring的BeanFactoryPostProcessor后置处理器，该处理器中设有自定义处理逻辑，用以将代理对

象注入到spring所管理的context中，供调用者通过spring注解机制获取代理对象，通过代理对象以thrift协议触发对远程服务的调用。

[0044] 还包括服务端模块，所述包括基于thrift协议的若干预设服务，所述服务端模块可根据调用请求对应的代理对象，向调用者反馈对应的预设服务，并记录执行数据，所述执行数据包括输入流、输出流、基于thrift协议的预设服务、调用成功情况、响应时间中的一种或多种。

[0045] 为进一步说明本发明所述基于Thrift协议的远程过程调用系统的工作流程和实现方式，以下以Java环境为例，对基于Thrift协议的远程过程调用系统调用流程进行说明：

[0046] 如图2所示：

[0047] 1、假设有一个已经存在的Thrift中间描述文件-UserService.thrift，并在其中定义了一些服务，如UserService；

[0048] 2、定义一个注解@STFService，里面定义url和tprotocol两个变量，url用于定义一个服务的远程地址，tprotocol用于定义这个远程服务所以使用的Thrift网络数据传输格式；

[0049] 3、产生的注解会存放于一个公共的基础包中，我们称为soeasy-thrift-commons；

[0050] 4、约定在开发任何一个服务前，先定义一个接口并约定名称以大写‘I’打头，后面接在中间描述文件中定义的服务名，并继承对应服务的Iface接口，如IUserService，并在接口上打上注解@STFService定义好url和tprotocol，暂称之为约定接口；

[0051] 5、利用Thrift编译器和thrift-maven-plugin插件，约定接口与对应的thrift中间描述文件所生成的thrift调用基础类打成一个包，我们称这个包为API包，命名依据具体服务场景而定；

[0052] 6、创建一个新包，定义一些基础工具类，这些工具类可以利用java反射原理读取任何API包中约定格式的接口，并提取出该接口对应的服务的url和网络传输格式tprotocol，结合API包中thrift编译器生成的基础代码，利用java的动态代理技术，根据约定接口动态生成一个代理对象，业务开发者拿到动态代理对象就可以像使用任何本地对象一样，我把这个工具类新包成为TFClientFactory；动态生成代理对象通过以下方式实现：利用Spring的BeanFactoryPostProcessor，添加逻辑使引入了API包和TFClientFactory包的工程可以在spring容器启动后自动扫描所有该工程引入的约定接口，并利用TFClientFactory根据扫描到的约定接口生成对应的代理对象，并将产生的代理对象存入spring context，这样在业务服务中，直接在约定接口上打@Autowired注解就会自动注入代理对象，就像使用本地service一样进行远程调用；

[0053] 7、在前述步骤的基础上，在代理方法被执行时利用Hystrix进行包装，收集对应服务对应方法的执行数据，包括成功次数，失败次数，失败比例，调用时长等数据，并以一定频率(如:10s一次)将统计数据发往时序存储数据库influxDB，然后通过grafana实时将数据以图表形式展示出来；

[0054] 8、定义一个新的工具包，用于服务端处理。主要入口需提供三个参数：输入流、输出流、thrift服务具体实现的实例，处理逻辑读取API包中的通信格式tprotocol，并结合thrift编译器生成的基础类以及thrift服务具体实现实例将输入流读取为请求对象，并调用具体实现实例进行处理，处理结果回写入输出流，完成一次调用。我们将这个新包称为

TFSFactory。

[0055] 9、在前述步骤的基础上，扩展默认的TProtocol，并实现一个自定义的TProcessor，在扩展的TProtocol中再解析传输协议时增加详细的日志监控，包括服务端的调用性能，传递的参数等数据。

[0056] 为进一步说明本发明的技术方案，以定时调度系统对外提供的服务为例，本实施例采用HTTP通讯协议，并基于Nginx构架，如图3所示。

[0057] 1、创建API包，添加中间描述文件QuartzService.thrift，定义TQuartzService，利用Thrift编译器变出Thrift基础代码，创建接口IQuartzService继承TQuartzService.Iface，添加注解@STFService(url＝"http://xxx.souyidai.com/quartz/xxx",tprotocol＝TBinaryProtocol.class)定义好服务的地址以及网络传输所采用的数据格式。

[0058] 2、Maven打包发布API

[0059] 3、实现服务端逻辑，并使用TFSFactory在web controller中开启Thrift服务

[0060] 4、调用方如果想调用quartz的服务，仅需要引入quartz服务的API包以及公共的TFCFactory包，并在需要使用的地方直接@Autowired即可，代码量对比可以参考图4，图中上半部分为使用原生thrift方式呈现的代码量，下半部分属于使用本发明所述技术方案呈现的代码量；

[0061] 通过图4可以看出，本发明所述技术方案完全屏蔽掉了Thrift的代码，使得业务代码更为清晰简洁，易用；

[0062] 由于采用http协议，当监控到请求量很大的情况下，可以直接通过nginx代理服务器进行流量调度；

[0063] 调用统计数据直接在grafana中实时显示。可以监控调用量，服务的健康情况，以便随时调整调度策略。

图1

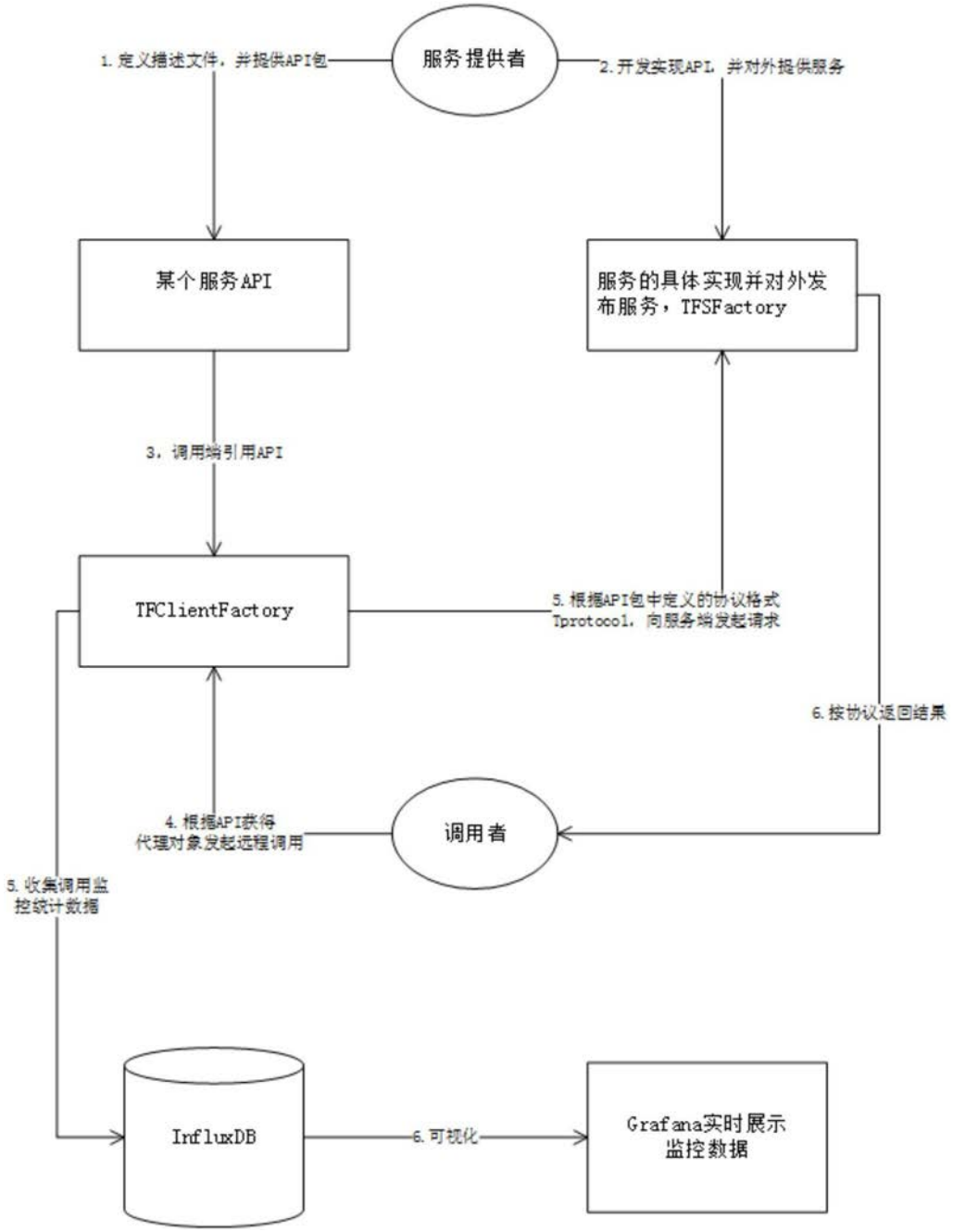
服务提供者
1.定义描述文件，并提供API包
2.开发实现API，并对外提供服务
某个服务API
服务的具体实现并对外发布服务，TFSFactory
3，调用端引用API
TFClientFactory
5.根据API包中定义的协议格式Tprotocol，向服务端发起请求
6.按协议返回结果
4.根据API获得代理对象发起远程调用
调用者
5.收集调用监控统计数据
InfluxDB
6.可视化
Grafana实时展示监控数据

图2

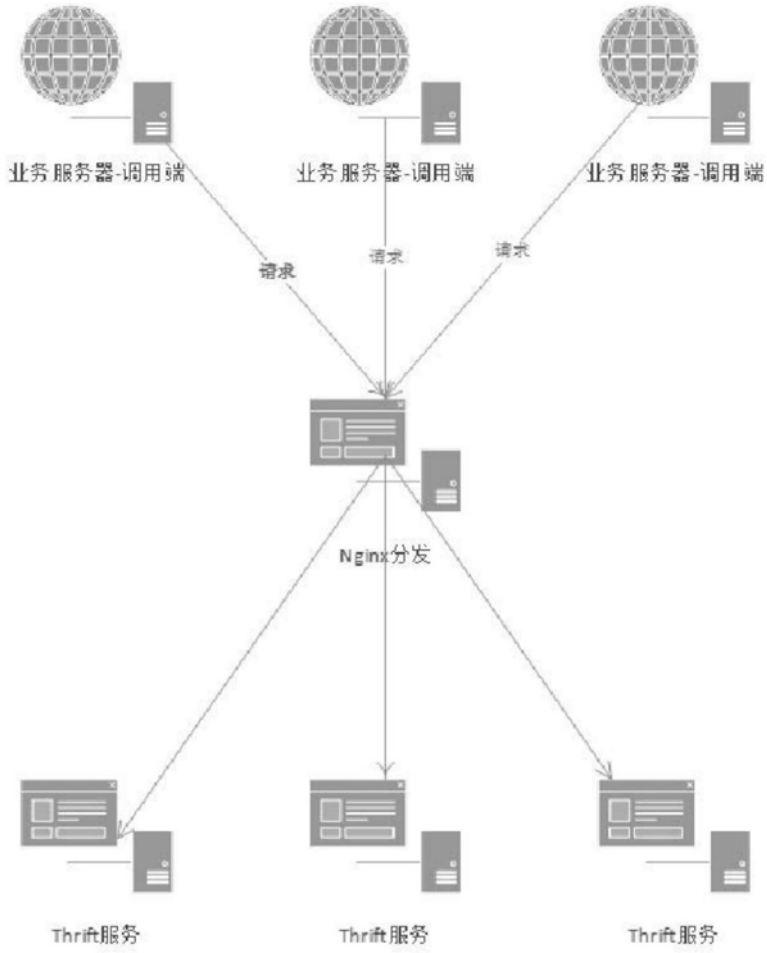

图3

```
HttpClient httpClient = HttpClientFactory.create(10000);
THttpClient thc = new THttpClient("http://xxx.souyidai.com/quartz/xxx",
httpClient);
TProtocol tp = new TBinaryProtocol(thc);
TQuartzService.Client client = new TQuartzService.Client(tp);
client.delJob("test");
thc.close();
```

```
@Autowired
private IQuartzService quartzService;
```

图4