(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111309374 B
(45)授权公告日 2022.11.01

(21)申请号 202010071580.9

(22)申请日 2020.01.21

(65)同一申请的已公布的文献号
申请公布号 CN 111309374 A

(43)申请公布日 2020.06.19

(73)专利权人 苏州达家迎信息技术有限公司
地址 215300 江苏省苏州市昆山开发区柏庐南路1001号博悦广场2区27号-129号

(72)发明人 陈强松

(74)专利代理机构 北京品源专利代理有限公司 11332
专利代理师 孟金喆

(51)Int.Cl.
G06F 8/71(2018.01)
G06F 8/36(2018.01)
H04L 41/5041(2022.01)

(56)对比文件
CN 107295077 A,2017.10.24
CN 107786379 A,2018.03.09
CN 108874567 A,2018.11.23
CN 106487594 A,2017.03.08
US 2019173940 A1,2019.06.06

审查员 殷娉

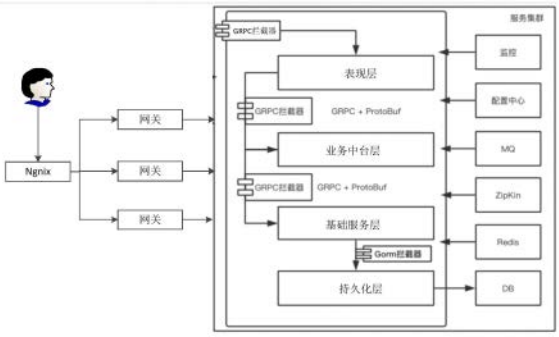
权利要求书3页 说明书9页 附图1页

(54)发明名称

一种微服务系统和微服务系统中的服务调用方法

(57)摘要

本发明实施例公开了一种微服务系统和微服务系统中的服务调用方法。系统包括:业务表现层、业务中台层和基础服务层;表现层用于通过对外暴露的接口接收用户的请求消息,并根据用户的请求消息调用所述业务中台层对外暴露的接口和/或所述基础服务层对外暴露的接口;业务中台层对所述用户的请求消息进行业务处理,并根据业务处理结果调用所述基础服务层对外暴露的接口;基础服务层用于访问持久化层,对所述请求消息对应的数据和/或业务处理结果对应的数据进行处理。本实施例通过调用接口的方式将请求消息不断向下传递,提高程序的可维护性,减少冗余代码。



CN 111309374 B

1.一种微服务系统,其特征在于,包括:表现层、业务中台层和基础服务层;

所述表现层,用于通过对外暴露的接口接收用户的请求消息,并根据所述用户的请求消息调用所述业务中台层对外暴露的接口和/或所述基础服务层对外暴露的接口;

所述业务中台层,用于对所述用户的请求消息进行业务处理,并根据业务处理结果调用所述基础服务层对外暴露的接口;

所述基础服务层,用于访问持久化层,对所述请求消息对应的数据和/或业务处理结果对应的数据进行处理;

所述表现层对外暴露的接口为实际业务场景的接口,所述业务中台层对外暴露的接口为对实际业务场景进行抽象化的抽象业务场景的接口,所述基础服务层对外暴露的接口为多种所述实际业务场景或多种抽象业务场景对应的数据操作类型的接口;所述表现层对外暴露的接口数量大于所述业务中台层对外暴露的接口数量;所述业务中台层对外暴露的接口数量大于所述基础服务层对外暴露的接口数量。

2.根据权利要求1所述的系统,其特征在于,

所述表现层、业务中台层和基础服务层之间,采用远程过程调用进行网络连接;

所述表现层、业务中台层和基础服务层之间,传输通用的传输对象,所述通用的传输对象用于传输通用的业务对象至调用的接口。

3.根据权利要求2所述的系统,其特征在于,

所述通用的传输对象为数据结构对象,并采用压缩比大于预设阈值的编码方法进行编码传输。

4.根据权利要求2所述的系统,其特征在于,还包括:对象生成工具;

所述对象生成工具,用于根据业务数据库、与所述业务数据库匹配的远端接口和对象关系映射框架,自动生成通用的业务对象。

5.根据权利要求1所述的系统,其特征在于,还包括:客户端拦截器和服务端拦截器;

所述客户端拦截器,用于对所述用户的请求消息进行解析,得到所述请求消息对应的租户标识;

所述服务端拦截器,用于拦截上层的请求消息;从所述客户端拦截器读取所述请求消息对应的租户标识;采用所述租户标识对所述请求消息进行处理。

6.根据权利要求5所述的系统,其特征在于,所述服务端拦截器包括:分别设置在网关与表现层之间、所述表现层与业务中台层之间,以及所述业务中台层与基础服务层之间的远程过程调用拦截器,和设置在所述基础服务层与所述持久化层之间的对象关系映射拦截器;

所述远程过程调用拦截器,用于截取上层的请求消息,对所述请求消息添加对应的租户标识,生成新的请求消息;并将所述新的请求消息发送至下层;

所述对象关系映射拦截器,用于在根据请求消息操作数据库时,对操作的数据添加所述请求消息对应的租户标识。

7.根据权利要求5所述的系统,其特征在于,

所述服务端拦截器,还用于运行系统监控报警框架,对监控的数据进行上报;

所述服务端拦截器,还用于运行分布式跟踪系统,对所述微服务系统中的调用链和服务性能进行监控。

8.根据权利要求1-7任一项所述的系统，其特征在于，还包括：与所述持久化层连接的中间件；

所述基础服务层，还用于配置基础服务层中各模块对数据表群的访问权限；

所述中间件，用于将所述基础服务层中模块的访问请求发送至所述持久化层中授权访问的数据表群。

9.根据权利要求1-7任一项所述的系统，其特征在于，还包括：配置中心；

所述配置中心用于：如果配置信息被更新，将更新后的配置信息发送至订阅的服务进程。

10.根据权利要求1-7任一项所述的系统，其特征在于，还包括：配置于基础服务层和持久化层之间的缓存层；

所述基础服务层，具体用于访问缓存层，对所述请求消息对应的数据和/或业务处理结果对应的数据进行处理；

所述缓存层，用于如果不包括所述请求消息对应的数据和/或业务处理结果对应的数据，访问持久化层读取所述对应的数据，对所述读取的数据进行处理。

11.一种微服务系统中的服务调用方法，其特征在于，所述微服务系统包括：表现层、业务中台层和基础服务层；所述方法包括：

通过所述表现层通过对外暴露的接口接收用户的请求消息，并根据所述用户的请求消息调用所述业务中台层对外暴露的接口和/或所述基础服务层对外暴露的接口；

通过业务中台层对所述用户的请求消息进行业务处理，并根据业务处理结果调用所述基础服务层对外暴露的接口；

通过所述基础服务层访问持久化层，对所述请求消息对应的数据和/或业务处理结果对应的数据进行处理；所述表现层对外暴露的接口为实际业务场景的接口，所述业务中台层对外暴露的接口为对实际业务场景进行抽象化的抽象业务场景的接口，所述基础服务层对外暴露的接口为多种所述实际业务场景或多种抽象业务场景对应的数据操作类型的接口；所述表现层对外暴露的接口数量大于所述业务中台层对外暴露的接口数量；所述业务中台层对外暴露的接口数量大于所述基础服务层对外暴露的接口数量。

12.根据权利要求11所述的方法，其特征在于，

所述表现层、业务中台层和基础服务层之间，采用远程过程调用进行网络连接；

所述表现层、业务中台层和基础服务层之间，传输通用的传输对象，所述通用的传输对象用于传输通用的业务对象至调用的接口。

13.根据权利要求11所述的方法，其特征在于，在所述通过所述表现层通过对外暴露的接口接收用户的请求消息之前，还包括：

通过客户端拦截器对用户的请求消息进行解析，得到请求消息对应的租户标识；

在所述通过客户端拦截器对用户的请求消息进行解析，得到请求消息对应的租户标识之后，所述方法还包括：

通过服务端拦截器拦截上层的请求消息；

从所述客户端拦截器读取所述请求消息对应的租户标识；

采用所述租户标识对所述请求消息进行处理。

14.根据权利要求11所述的方法，其特征在于，所述方法还包括以下操作中的至少一

种：

通过基础服务层配置基础服务层中各模块对数据表群的访问权限，并通过中间件将所述基础服务层中模块的访问请求发送至所述持久化层中授权访问的数据表群；

通过配置中心判定配置信息被更新，将更新后的配置信息发送至订阅的服务进程；

通过基础服务层访问缓存层，对所述请求消息对应的数据和/或业务处理结果对应的数据进行处理，并通过所述缓存层判定不包括所述请求消息对应的数据和/或业务处理结果对应的数据，访问持久化层读取所述对应的数据，对所述读取的数据进行处理。

一种微服务系统和微服务系统中的服务调用方法

技术领域

[0001] 本发明实施例涉及计算机技术领域，尤其涉及一种微服务系统和微服务系统中的服务调用方法。

背景技术

[0002] 微服务最早由Martin Fowler与James Lewis于2014年共同提出，微服务架构风格是一种使用一套小服务来开发单个应用的方式途径，每个服务运行在自己的进程中，并使用轻量级机制通信。

[0003] 目前，一个大型复杂软件应用由一个或多个微服务组成。系统中的各个微服务可被独立部署，各个微服务之间是松耦合的。每个微服务仅关注于完成一件任务并很好地完成该任务。在对用户的请求进行响应时，需要通过微服务之间的接口调用，执行对应的微服务。

[0004] 在实际业务场景中，由于业务常常变化，微服务的调用接口需要重新定义，导致很多的接口定义重复，出现了很多功能类似的接口；而且，大量的重复代码带来了严重的代码维护问题，经常出现代码漏改的问题。

发明内容

[0005] 本发明实施例提供一种微服务系统和微服务系统中的服务调用方法，以提高程序的可维护性，减少冗余代码。

[0006] 第一方面，本发明实施例提供了一种微服务系统，其特征在于，包括：表现层、业务中台层和基础服务层；

[0007] 所述表现层，用于通过对外暴露的接口接收用户的请求消息，并根据所述用户的请求消息调用所述业务中台层对外暴露的接口和/或所述基础服务层对外暴露的接口；

[0008] 所述业务中台层，用于对所述用户的请求消息进行业务处理，并根据业务处理结果调用所述基础服务层对外暴露的接口；

[0009] 所述基础服务层，用于访问持久化层，对所述请求消息对应的数据和/或业务处理结果对应的数据进行处理。

[0010] 本发明实施例提供的微服务系统包括表现层、业务中台层和基础服务层，分层级对系统进行控制，简化微服务系统的复杂度，规范设计过程，加快项目的交付速度；表现层可以调用业务中台层，也可以跨层调用基础服务层，满足多样化的业务需求；各层之间采用预先定义的接口进行交互，通过调用接口的方式将请求消息不断向下传递；当业务发生变化时，本实施例提供的接口调用方式依然不变，无需增加接口，也无需修改接口定义，提高程序的可维护性，减少冗余代码。

[0011] 第二方面，本发明实施例还提供了一种微服务系统中的服务调用方法，所述微服务系统包括：表现层、业务中台层和基础服务层；所述方法包括：

[0012] 通过所述表现层通过对外暴露的接口接收用户的请求消息，并根据所述用户的请

求消息调用所述业务中台层对外暴露的接口和/或所述基础服务层对外暴露的接口；

[0013] 通过业务中台层对所述用户的请求消息进行业务处理，并根据业务处理结果调用所述基础服务层对外暴露的接口；

[0014] 通过所述基础服务层访问持久化层，对所述请求消息对应的数据和/或业务处理结果对应的数据进行处理。

附图说明

[0015] 图1为本发明实施例一提供的微服务系统的架构图；

[0016] 图2为本发明实施例五提供的一种微服务系统中的服务调用方法的流程图。

具体实施方式

[0017] 下面结合附图和实施例对本发明作进一步的详细说明。可以理解的是，此处所描述的具体实施例仅仅用于解释本发明，而非对本发明的限定。另外还需要说明的是，为了便于描述，附图中仅示出了与本发明相关的部分而非全部结构。

[0018] 实施例一

[0019] 图1为本发明实施例一提供的微服务系统的架构图，相比于现有包括多个服务的微服务框架，本实施例提供的系统为领域驱动设计下的3层结构，按照由上至下的顺序分别为表现层(Present层)、业务中台层(Bmp层)、基础服务层(Basic层)。其中，表现层、业务中台层和基础服务层实质为通过相应的逻辑代码实现的单元，每个单元包括多个业务模块，用于实现对应的功能。

[0020] 其中，表现层面向用户和界面(UI)，负责和用户进行交互，包括全球广域网(World Wide Web，Web)页面、手机软件页面、供第三方调用的接口等，适应灵活多变的业务体系。业务中台层主要用来处理对业务的封装和沉淀，主要面向对业务的沉淀和积累。业务中台层还包括异步的消息队列(MQ)消费者。在高并发分布式环境下，由于来不及同步处理，请求往往发生堵塞，比如说，大量的insert、update之类的请求同时到达数据库，直接导致无所的行锁和表锁，甚至最后请求会堆积过多，从而触发很多连接错误。通过使用MQ，消息从队列推送至消费者后，消息被消费，并从队列中移除。这样可以异步处理请求，从而缓解系统的压力。基础服务层面向持久化层，主要用来作数据的存储和更新保存，面向数据结构。持久化层包括数据库(DB)。

[0021] 可选地，对于手机软件或者使用超文本标记语言(HyperText Markup Language，HTML)直接实现的客户端，或者是.net实现的桌面客户端，由于这些客户端是单独开发的，没有控制层，因此需要增加一层“应用程序编程接口(Application Programming Interface，API)网关”(以下简称网关)作为这些客户端的控制层，以便接入微服务系统。如图1所示，微服务系统还包括网关，作为系统的统一入口，外部通过统一的网关接入微服务系统，同时处理一些功能。

[0022] 可选地，网关处理的功能包括但不限于：授权、接口权限控制和限流。其中，网关增加了基于一种开放的协议(Oauth)的4种授权模式：授权码模式，密码模式，简化模式，客户端凭证。接口权限控制可以分为三个部分：用户认证，服务权限，用户权限。限流可以保障API服务对所有用户的可用性，也可以防止网络攻击。限流策略包括：限制总并发数(比如数

据库连接池、线程池)、限制瞬时并发数、限制时间窗口内的平均速率(如Guava的RateLimiter、nginx的limit_req模块,限制每秒的平均速率);其他还有如限制远程接口调用速率、限制MQ的消费速率。另外还可以根据网络连接数、网络流量、CPU或内存负载等来限流。

[0023] 本实施例中,表现层、业务中台层和基础服务层分别对外暴露对应的接口。具体的,表现层对外暴露的接口为实际业务场景的接口,业务中台层对外暴露的接口为对实际业务场景进行抽象化的抽象业务场景的接口,基础服务层对外暴露的接口为多种实际业务场景或多种抽象业务场景对应的数据操作类型的接口。在一示例中,实际业务场景的接口包括手机号的接口、用户名的接口、密码的接口、变更密码的接口和变更昵称的接口等。对手机号的接口、用户名的接口和密码进行抽象化,得到注册场景的接口,对变更密码和变更昵称进行抽象化,得到变更用户信息的接口。虽然业务场景有很多,但是多种业务场景对应的数据操作类型是有限的,例如增加数据、更改数据等。在微服务系统支持多租户的应用场景中,基础服务层对外暴露的接口包括:根据租户标识查询租户的接口、根据租户标识集合查询租户集合的接口、根据租户名称模糊查询租户集合、根据条件查询租户集合、业务下拉接口等。

[0024] 可以理解的是,由于业务中台层对实际业务场景进行了抽象,则表现层对外暴露的接口数量大于业务中台层对外暴露的接口数量。业务中台层对外暴露的接口数量大于基础服务层对外暴露的接口数量,从而接口逐级收敛。

[0025] 基于各层对外暴露的接口,在微服务系统中进行服务调用时,用户通过客户端向微服务系统发送请求消息。请求消息经过Web服务器(如Nginx)后到达网关。网关接收该请求消息后,验证请求消息对应的用户是否被授权、是否具有接口权限。如果用户被授权且具有接口权限,将请求消息转发至表现层。表现层通过对外暴露的接口接收用户的请求消息,并根据所述用户的请求消息调用业务中台层对外暴露的接口和/或基础服务层对外暴露的接口。具体地,根据请求消息确定实际业务场景,调用与该实际业务场景对应的抽象业务场景的接口,和/或,根据请求消息调用对应的数据操作类型的接口。具体调用哪个接口由业务的具体内容决定。

[0026] 业务中台层,用于对用户的请求消息进行业务处理,并根据业务处理结果调用基础服务层对外暴露的接口。具体地,业务中台层包括多个业务模块,每个业务模块负责处理不同类型的业务,如订单业务、库存业务等。针对用户的请求消息确定至少一个业务模块,由至少一个业务模块单独或协同对请求消息进行业务处理,生成业务处理结果。然后,根据业务处理结果确定数据操作类型,进而调用基础服务层中该数据操作类型对应的接口。例如,订单业务模块对订单查询请求进行订单验证,包括订单编号验证和订单内容验证,生成验证通过或者不通过的业务处理结果,如果验证通过确定查询类型并调用查询接口。

[0027] 基础服务层用于访问持久化层,对请求消息对应的数据和/或业务处理结果对应的数据进行处理。例如,业务处理结果对应新的数据,如新注册的用户数据,则将该新的数据存储到数据库中;又例如,业务处理结果对应更新的数据,如用户更改的业务需求数据,则在数据库中对该更新的数据进行更新存储。

[0028] 本发明实施例提供的微服务系统包括表现层、业务中台层和基础服务层,分层级对系统进行控制,简化微服务系统的复杂度,规范设计过程,加快项目的交付速度;表现层

可以调用业务中台层，也可以跨层调用基础服务层，满足多样化的业务需求；各层之间采用预先定义的接口进行交互，通过调用接口的方式将请求消息不断向下传递；当业务发生变化时，本实施例提供的接口调用方式依然不变，无需增加接口，也无需修改接口定义，提高程序的可维护性，减少冗余代码。

[0029] 进一步地，按照从上到下的顺序，接口逐级收敛，保证接口的数量在一定范围内，同时提高底层接口的复用率。

[0030] 实施例二

[0031] 本实施例对上述实施例中各层级之间的传输方式进行进一步优化，以提高传输效率。

[0032] 本实施例中的分层体系建立在远程过程调用(Romote Procedure Call，RPC)的基础上，表现层、业务中台层和基础服务层之间，采用RPC进行网络连接。可选地，RPC可以是Google出品的gRPC。gRPC是一个高性能、通用的开源RPC框架，基于HTTP/2协议标准和Protobuf序列化协议开发，支持众多的开发语言。

[0033] 可选地，本实施例将请求消息封装为通用的传输对象；基于此，表现层、业务中台层和基础服务层之间，传输通用的传输对象。其中，通用的传输对象用于传输通用的业务对象至调用的接口。

[0034] 本实施例中，通用的传输对象为数据结构对象，例如CommonDao对象。数据结构对象的.proto描述文件内容如下所示：

[0035] Message xxx{

[0036] Int32 code＝1;

[0037] String msg＝2;

[0038] Google.protobuf.Any detail＝3;

[0039] }

[0040] 可见，数据结构对象包括指定字段、指定字段的描述和通用的业务对象(protobuf Any)；采用protobuf Any对多类型的对象进行统一支持，既保证了编码的高效性，也保证了扩展性。通用的业务对象封装请求消息的主体内容。在一示例中，用户发送某个租户标识的查询请求消息。表现层对该查询请求消息进行验证后，生成通用的传输对象并发送至对应的接口。该通用的传输对象中指定字段为ok，描述为验证通过，通用的业务对象对租户标识。

[0041] 要让通用的传输对象能在网络上传输或存储，需要对通用的传输对象进行编码和解码。可选地，采用压缩比大于预设阈值的编码方法进行编码传输，预设阈值可以自主设定，例如1/3。可选地，采用protobuf对通用的传输对象进行编码传输。protobuf是一个灵活、高效、结构化的数据序列化框架，具有高的压缩比，兼容好等优点；相比于xml等传统的序列化工具，它更小，更快，更简单。protobuf支持数据结构化一次可以到处使用，甚至跨语言使用，通过代码生成工具可以自动生成不同语言版本的源代码，甚至可以在使用不同版本的数据结构进程间进行数据传递，实现数据结构的前向兼容。采用protobuf编码后，表现层与业务中台层之间，以及业务中台层与基础服务层之间的传输格式为protobuf。使用protobuf对整个传输对象进行数据传输，提高程序执行效率，提高服务访问效率。

[0042] 为了减少编程工作量以及对象定义开销的时间，本实施例提供了自动生成通用的

业务对象的对象生成工具。对象生成工具用于根据业务数据库、与业务数据库匹配的远端接口和对象关系映射框架，自动生成通用的业务对象。生成的业务对象支持protobuf格式，可以在各层间进行传输。

[0043] 实施例三

[0044] 本实施例对上述实施例进行进一步优化，在微服务系统中实现租户体系支持。

[0045] 本实施例中的微服务系统需要针对不同的用户分配不同的资源区域，符合租户技术的设定，多租户技术(multi-tenancy technology)，是一种软件架构技术，它是在探讨与实现如何于多用户的环境下共用相同的系统或程序组件，并且仍可确保各用户间数据的隔离性。本实施例在网关中开发了对租户(Tenant)的支持。具体地，网关对请求消息进行租户身份认证，方法包括令牌(Token)认证、访问密钥(Access Key Id，AK)/密钥(Secret Access Key，SK)认证和非对称加密方式认证。以安全性更高的AK/SK为例，AK用于标示用户，SK是用户用于加密认证字符串和网关用来验证认证字符串的密钥。每个租户分配有不同的AK和SK用来进行访问。请求消息包括AK、请求内容和使用SK计算的签名。

[0046] 网关配置了传递租户标识的客户端拦截器，用于对用户的请求消息进行解析，得到请求消息对应的租户标识。具体地，客户端拦截器，用于根据请求消息中的AK找到对应的SK，使用与客户端同样的算法将请求内容和SK一起计算签名。对比客户端发送的签名和计算得到的签名，若一致则身份验证通过，否则身份验证失败。当身份验证通过时，将AK或者AK对应的字符串作为租户标识。接着，客户端拦截器将租户标识放入客户端进程中的上下文对象(context)中，用来进行传递；然后从上下文对象中读取租户标识并放入请求桶中。

[0047] 对于服务端拦截器，拦截上层的请求消息；从客户端拦截器读取请求消息对应的租户标识；采用租户标识对请求消息进行处理，以便按照租户标识隔离请求消息。具体地，如图1所示，服务端拦截器包括：分别设置在网关与表现层之间、表现层与业务中台层之间，以及业务中台层与基础服务层之间的远程过程调用拦截器，和设置在基础服务层与持久化层之间的对象关系映射(orm)拦截器。

[0048] 其中，在采用gRPC实现层级之间的网络连接的基础上，可继续采用gRPC拦截器。其用于把租户标识一层一层地往下进行传递，最后一直传递到基础服务层。基础服务层与持久层之间实现了orm(例如Google出品的gorm)拦截器。其中，gRPC的拦截器截取上层的请求消息，对请求消息添加对应的租户标识，生成新的请求消息；并将新的请求消息发送至下层。其中，租户标识可以从请求桶中读取，并将读取的租户标识放入服务端进程中的上下文对象中，从而实现添加租户标识，隔离不同租户的请求消息。可见，新的请求消息包括截取到的请求消息和附属的租户标识。值得说明的是，对于设置在网关与表现层之间的gRPC拦截器来说，上层为网关。

[0049] gorm拦截器可以直接从服务端进程中的上下文对象中读取租户标识。在根据请求消息操作数据库时，对操作的数据添加请求消息对应的租户标识。其中，操作的数据包括增加、修改、查询或删除的数据，从而保证单个租户的数据独立性，单个租户的数据只能被自己访问，只能被自己修改。

[0050] 可选地，gRPC的拦截器采用责任链模式(ServerInterceptorChain)实现。责任链模式的最大作用就是可以让请求消息在链上传播，那么链上每个节点都可以对请求消息进行处理。这里的请求是一次rpc调用，包含调用的方法名，调用方法参数列表、参数值、调用

方法所属的类等等信息。具体地，责任链上的每个节点各自对请求消息添加租户标识、数据上报和对调用链和服务性能进行监控。

[0051] 对于数据上报，在gRPC的拦截器中搭建系统监控报警框架，例如Prometheus，运行系统监控报警框架对监控的数据进行上报。例如，Prometheus是通过HTTP协议周期性抓取被监控组件的状态，任意组件只要提供对应的HTTP接口就可以接入监控。

[0052] 对于对调用链和服务性能进行监控，在gRPC的拦截器中搭建分布式跟踪系统，例如Zipkin，运行分布式跟踪系统对微服务系统中的调用链和服务性能进行监控。例如，运行分布式跟踪系统在服务调用的请求和响应中加入标识，标明上下游请求的关系。利用这些信息，可以可视化地分析服务调用链路和服务间的依赖关系，从而对调用链和服务性能进行监控。

[0053] 本实施例中，通过数据上报以及对调用链和服务性能进行监控，方便业务开发，提高开发定位问题的能力。数据上报和对调用链和服务性能进行监控部署在拦截器中，对业务开发无感知。本实施例中，通过拦截器实现切面编程，提高程序的可扩展性。

[0054] 实施例四

[0055] 本实施例对上述实施例进行进一步优化，在微服务系统中增加与持久层连接的中间件，隔离持久层和基础服务层之间的强耦合。

[0056] 具体地，基础服务层包括至少两个实现，分别处理不同的业务，则两个模块需要访问的数据表也不同。为了避免持久层访问错误，基础服务层用于配置基础服务层中各模块对数据表群的访问权限。其中，数据表群由至少一张数据表构成。例如，基础服务层包括用户信息的基础服务层实现和订单信息的基础服务层实现，配置用户信息的基础服务层实现对用户数据表群的访问权限，配置订单信息的基础服务层实现对订单数据表群的访问权限。

[0057] 中间件将基础服务层中模块的访问请求发送至持久层中授权访问的数据表群。中间件例如是KingShard，KingShard是一个Go开发的mysql中间件，可以实现读写分离、分库分表、连接池等功能。本实施例通过KingShard和数据库进行连接，中间等于加了一次代理，方便后续进行数据库的水平拆分和垂直拆分。可选地，在数据库水平切分或者垂直拆分后，通过中间件直接将基础服务层中模块的访问请求发送至持久层中对应的切分后的数据表中，有利于提高业务的吞吐量。

[0058] 在一示例中，用户信息的基础服务层实现和订单信息的基础服务层实现均具有对数据库A的访问权限，数据库A包括用户数据表和订单数据表。对数据库A按照业务类型进行垂直切分，将用户数据表和订单数据表分布到数据库A和数据库B上。此时，用户信息的基础服务层实现和订单信息的基础服务层实现均具有数据库A和数据库B的访问权限。与数据库A和数据库B连接的中间件，接收各实现发送的访问请求，根据访问请求的业务类型发送至对应的数据库。具体地，将用户信息的基础服务层实现发送的访问请求发送到数据库A，将订单信息的基础服务层实现发送的访问请求发送到数据库B。

[0059] 在另一示例中，用户信息的基础服务层实现具有对数据表C的访问权限，当用户信息激增的时候，一个数据表容量不够，需要扩展到多个数据表。对数据表C进行水平切分，将数据表C拆分到数据库C和数据库D中。此时，用户信息的基础服务层实现具有数据库C和数据库D的访问权限。与数据库C和数据库D连接的中间件，接收用户信息的基础服务层实现发

送的访问请求，并将访问请求发送至数据库C和数据库D。

[0060] 本实施例通过对基础服务层进行严格的表群控制，保留了今后进行垂直拆分和水平拆分的扩展性，不同的业务可以部署在不同的数据库实例里，保证各项业务可以充分解耦。

[0061] 可选地，如图1所示，微服务系统还包括配置中心。配置中心通过配置信息对微服务系统中的各服务进程进行集中管理。配置中心可以根据微服务的运行情况自动更新配置信息，也可以接收用户的配置指令，根据配置指令更新配置信息。微服务系统中的各服务进程预先向配置中心订阅更新的配置信息。如果配置信息被更新，将更新后的配置信息发送至订阅的服务进程。配置中心的订阅模式能够适应灵活多变的业务体系，保证服务进程更新的时效性和自动化。

[0062] 可选地，如图1所示，微服务系统还包括缓存层(Redis)，缓存层配置于基础服务层和持久化层之间。基础服务层优先访问缓存层，如果缓存层不包括请求消息对应的数据和/或业务处理结果对应的数据，则缓存层访问持久化层读取对应的数据并进行处理；如果缓存层存在请求消息对应的数据和/或业务处理结果对应的数据，则由缓存层直接对对应的数据进行处理。进一步地，缓存层在对数据进行处理之后，将处理结果更新到持久层。进一步地，缓存层从持久化层读取数据后，会存储数据一段时间，以供后续服务调用时可以直接由缓存层进行处理。

[0063] 实施例五

[0064] 图2为本发明实施例五提供的一种微服务系统中的服务调用方法的流程图，适用于层级结构的微服务系统。微服务系统包括：表现层、业务中台层和基础服务层，对于微服务系统的描述详见上述各实施例，此处不再赘述。

[0065] 如图2的微服务系统中的服务调用方法，包括：

[0066] S210、通过表现层通过对外暴露的接口接收用户的请求消息，并根据所述用户的请求消息调用业务中台层对外暴露的接口和/或基础服务层对外暴露的接口。

[0067] S220、通过业务中台层对用户的请求消息进行业务处理，并根据业务处理结果调用基础服务层对外暴露的接口。

[0068] S230、通过基础服务层访问持久化层，对请求消息对应的数据和/或业务处理结果对应的数据进行处理。

[0069] 本发明实施例中，表现层可以调用业务中台层，也可以跨层调用基础服务层，满足多样化的业务需求；各层之间采用预先定义的接口进行交互，通过调用接口的方式将请求消息不断向下传递；当业务发生变化时，本实施例提供的接口调用方式依然不变，无需增加接口，也无需修改接口定义，提高程序的可维护性，减少冗余代码。

[0070] 可选地，表现层对外暴露的接口为实际业务场景的接口，业务中台层对外暴露的接口为对实际业务场景进行抽象化的抽象业务场景的接口，基础服务层对外暴露的接口为多种实际业务场景或多种抽象业务场景对应的数据操作类型的接口。

[0071] 可选地，表现层、业务中台层和基础服务层之间，采用远程过程调用进行网络连接；表现层、业务中台层和基础服务层之间，传输通用的传输对象，通用的传输对象用于传输通用的业务对象至调用的接口。

[0072] 可选地，通用的传输对象为数据结构对象，并采用压缩比大于预设阈值的编码方

法进行编码传输。

[0073] 可选地，在S210之前，上述方法还包括：通过对象生成工具根据业务数据库、与业务数据库匹配的远端接口和对象关系映射框架，自动生成通用的业务对象。

[0074] 可选地，在S210之前，上述方法还包括：通过客户端拦截器对用户的请求消息进行解析，得到请求消息对应的租户标识。进一步地，在通过客户端拦截器对用户的请求消息进行解析，得到请求消息对应的租户标识之后且S210之前，或者S210之后且S220之前，或者S220之后且S230之前，上述方法还包括：通过服务端拦截器拦截上层的请求消息；从客户端拦截器读取请求消息对应的租户标识；采用租户标识对请求消息进行处理。

[0075] 可选地，服务端拦截器包括：分别设置在网关与表现层之间、表现层与业务中台层之间，以及业务中台层与基础服务层之间的远程过程调用拦截器，和设置在基础服务层与持久化层之间的对象关系映射拦截器。包括：通过远程过程调用拦截器截取上层的请求消息，对请求消息添加对应的租户标识，生成新的请求消息；并将新的请求消息发送至下层；通过对象关系映射拦截器在根据请求消息操作数据库时，对操作的数据添加请求消息对应的租户标识。

[0076] 可选地，上述方法还包括：通过服务端拦截器运行系统监控报警框架，对监控的数据进行上报；通过服务端拦截器运行分布式跟踪系统，对微服务系统中的调用链和服务性能进行监控。

[0077] 可选地，上述方法还包括：通过基础服务层配置基础服务层中各模块对数据表群的访问权限；基于此，通过基础服务层访问持久化层包括：通过中间件将基础服务层中模块的访问请求发送至持久层中授权访问的数据表群。

[0078] 可选地，上述方法还包括：通过配置中心判定配置信息被更新，将更新后的配置信息发送至订阅的服务进程。

[0079] 可选地，通过基础服务层访问持久化层，对请求消息对应的数据和/或业务处理结果对应的数据进行处理，包括：通过基础服务层，访问缓存层，通过缓存层判定不包括请求消息对应的数据和/或业务处理结果对应的数据，访问持久化层读取对应的数据，对读取的数据进行处理。

[0080] 通过以上关于实施方式的描述，所属领域的技术人员可以清楚地了解到，本发明可借助软件及必需的通用硬件来实现，当然也可以通过硬件实现，但很多情况下前者是更佳的实施方式。基于这样的理解，本发明的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品可以存储在计算机可读存储介质中，如计算机的软盘、只读存储器(Read-Only Memory,ROM)、随机存取存储器(Random Access Memory,RAM)、闪存(FLASH)、硬盘或光盘等，包括若干指令用以使得一台计算机设备(可以是个人计算机，服务器，或者网络设备等)执行本发明各个实施例所述的方法。

[0081] 值得注意的是，上述系统的实施例中，所包括的各个单元和模块只是按照功能逻辑进行划分的，但并不局限于上述的划分，只要能够实现相应的功能即可；另外，各功能单元的具体名称也只是为了便于相互区分，并不用于限制本发明的保护范围。

[0082] 本发明中，系统实施例与方法实施例具有相对应的技术特征，方法实施例和系统实施例中的技术特征可以互相参照，不再赘述。

[0083] 注意，上述仅为本发明的较佳实施例及所运用技术原理。本领域技术人员会理解，

本发明不限于这里所述的特定实施例，对本领域技术人员来说能够进行各种明显的变化、重新调整和替代而不会脱离本发明的保护范围。因此，虽然通过以上实施例对本发明进行了较为详细的说明，但是本发明不仅仅限于以上实施例，在不脱离本发明构思的情况下，还可以包括更多其他等效实施例，而本发明的范围由所附的权利要求范围决定。

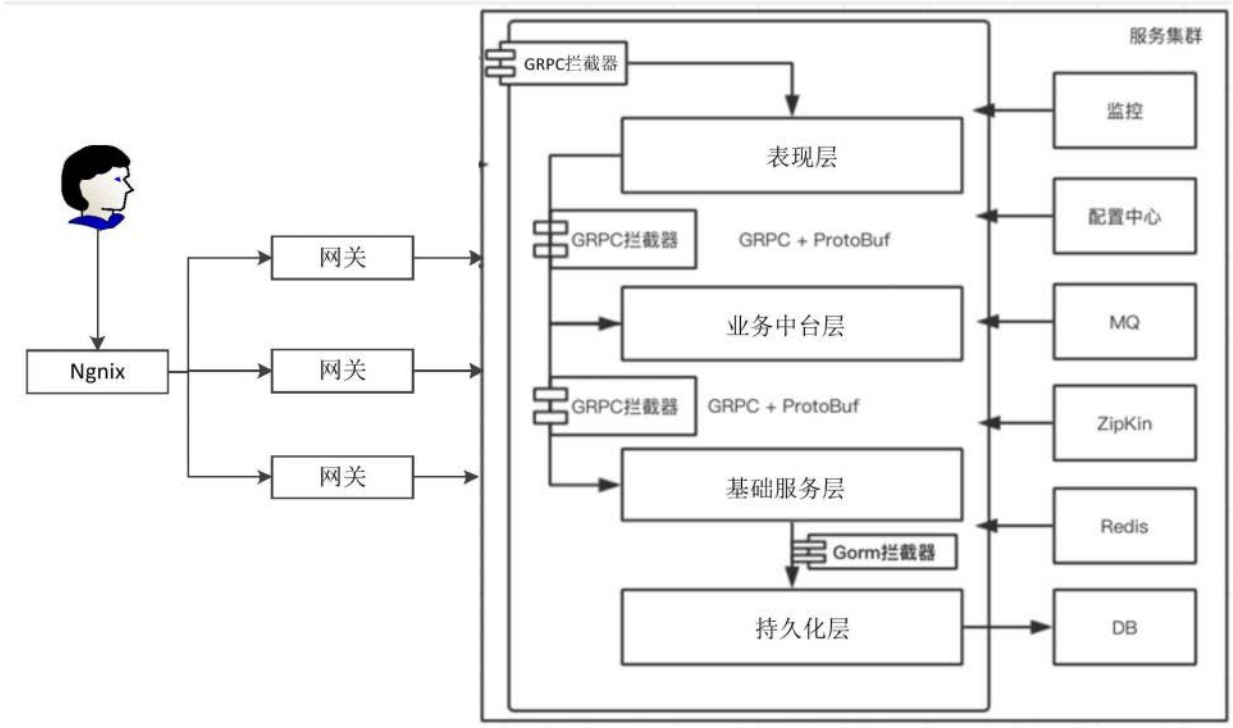

服务集群
GRPC拦截器
表现层
GRPC拦截器
GRPC + ProtoBuf
业务中台层
GRPC拦截器
GRPC + ProtoBuf
基础服务层
Gorm拦截器
持久化层
监控
配置中心
MQ
ZipKin
Redis
DB
网关
网关
网关
Ngnix

图1

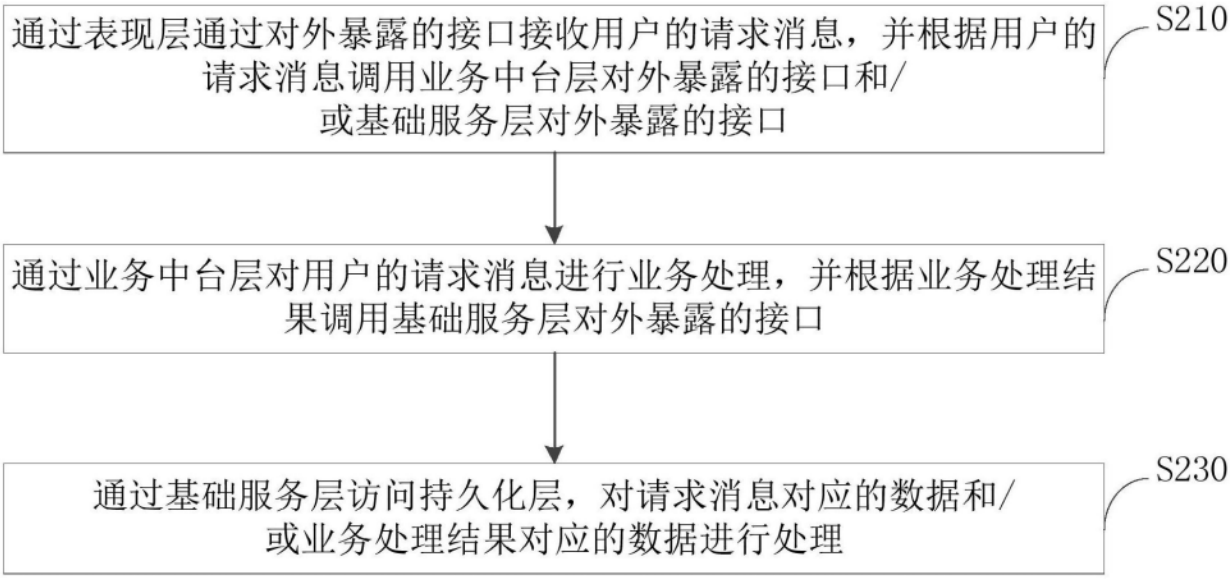

通过表现层通过对外暴露的接口接收用户的请求消息，并根据用户的请求消息调用业务中台层对外暴露的接口和/或基础服务层对外暴露的接口
S210
通过业务中台层对用户的请求消息进行业务处理，并根据业务处理结果调用基础服务层对外暴露的接口
S220
通过基础服务层访问持久化层，对请求消息对应的数据和/或业务处理结果对应的数据进行处理
S230

图2