



(12) 发明专利

(10) 授权公告号 CN 113722139 B

(45) 授权公告日 2024.08.20

(21) 申请号 202111000125.0

(56) 对比文件

(22) 申请日 2021.08.27

CN 102446123 A, 2012.05.09

(65) 同一申请的已公布的文献号

审查员 徐小婷

申请公布号 CN 113722139 A

(43) 申请公布日 2021.11.30

(73) 专利权人 东莞盟大集团有限公司

地址 523000 广东省东莞市南城街道黄金
路1号天安数码城4栋2单元1301室

专利权人 东莞市盟大数据科技有限公司

(72) 发明人 陈浩强

(74) 专利代理机构 广州三环专利商标代理有限
公司 44202

专利代理师 张艳美 赵月芬

(51) Int. Cl.

G06F 11/07 (2006.01)

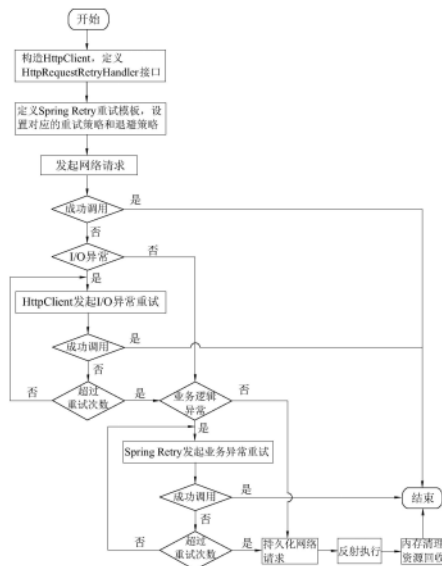
权利要求书1页 说明书4页 附图2页

(54) 发明名称

请求效率高且不易丢失数据的数据请求方
法

(57) 摘要

本发明公开了一种请求效率高且不易丢失数据的数据请求方法、执行该方法的电子设备以及存储该方法的计算机可读存储介质。本发明针对异常处理方面,划分为I/O异常、业务逻辑异常、以及I/O异常、业务逻辑异常之外的其它异常(非程序性异常)等多层面维度,对于业务逻辑异常,用Spring Retry处理,而对于I/O异常这种底层异常,则是采用HttpClient重试机制处理,利用HttpRequestRetryHandler异常重试接口能够更底层、更高效解决网络层面异常的问题,减小了程序损耗。且,在经HttpClient重试、Spring Retry重试没有实现成功调用以及非程序性异常情形,则是将网络请求包装成对象存储,以便下次重新触发完成网络请求,避免数据的丢失。



1. 一种请求效率高且不易丢失数据的数据请求方法,其特征在于,包括步骤:

S1,发起网络请求,以调用接口获取数据;

S2,获取接口调用结果,若成功调用,进入步骤S5,否则,进入步骤S3;

S3,确定所述网络请求对应的异常类型,若所述异常类型为I/O异常,采用HttpClient重试机制发起重试请求,若所述异常类型为业务逻辑异常,采用Spring Retry重试机制发起重试请求,经所述HttpClient重试和/或Spring Retry重试没有成功调用且已达到预先定义的重试阈值,或,所述异常类型为I/O异常、业务逻辑异常之外的其它异常,进入步骤S4;

S4,将所述网络请求包装成对象并存储;

S5,结束;

所述重试阈值包括第一重试阈值和第二重试阈值,在步骤S3中,首先判断所述异常类型是否为I/O异常,若是I/O异常,采用所述HttpClient重试机制发起重试请求,直至成功调用进入步骤S5或超过所述第一重试阈值;若超过所述第一重试阈值或所述异常类型不是I/O异常,判断所述异常类型是否为业务逻辑异常,若是业务逻辑异常,采用所述Spring Retry重试机制发起重试请求,直至成功调用进入步骤S5或超过所述第二重试阈值;若超过所述第二重试阈值或所述异常类型不是业务逻辑异常,进入步骤S4。

2. 如权利要求1所述的数据请求方法,其特征在于,在步骤S1之前,还包括:

S0,构造HttpClient,定义HttpRequestRetryHandler接口;以及,定义Spring Retry重试模板,设置对应的重试策略和退避策略。

3. 如权利要求2所述的数据请求方法,其特征在于,以单例模式构造HttpClient并设置超时时间,并利用PoolingHttpClientConnectionManager启用Http连接池实现连接复用。

4. 如权利要求1所述的数据请求方法,其特征在于,开启定时扫描空连接的单线程池任务,关闭异常或空闲连接。

5. 如权利要求4所述的数据请求方法,其特征在于,在JVM关闭时,执行addShutdownHook指令关闭HttpClient对象和关闭用于扫描空连接的单线程池。

6. 如权利要求1所述的数据请求方法,其特征在于,在步骤S4之后步骤S5之前,还包括:基于Java反射机制重新执行所述网络请求。

7. 一种电子设备,其特征在于,包括:

处理器、存储器以及存储在所述存储器中且被配置为由所述处理器执行的计算机程序,所述处理器执行所述计算机程序时,执行如权利要求1至6任一项所述的数据请求方法。

8. 一种计算机可读存储介质,其特征在于,所述计算机可读存储介质存储有计算机程序,所述计算机程序可被处理器执行以完成如权利要求1至6任一项所述的数据请求方法。

请求效率高且不易丢失数据的数据请求方法

技术领域

[0001] 本发明涉及数据交互技术领域,尤其涉及一种请求效率高且不易丢失数据的数据请求方法、电子设备及可读存储介质。

背景技术

[0002] 数据平台/仓库中,往往需要大量各行业纬度的数据来进行数据整合与分析、建模,从而提供有力的决策支持,所以源源不断从第三方数据源中获取数据是极有必要的。其中,通过接口形式调用较为常见,如何保证网络请求中能高效并不轻易丢失其中一条宝贵的数据是亟需去研究解决的问题。

[0003] 为提高网络发生异常时请求成功的概率,需要预先制定好重试机制,在请求调用失败之后重新尝试发起请求。现阶段,通常采用Spring Retry、Guava Retrying等重试组件实现异常重试处理,都实现了非常优秀的异常重试处理机制,但不涉及业务逻辑的IOException异常也抛出到业务层面重试处理,导致不必要的额外程序损耗。且,存在因短期网络不稳定或数据源方服务器宕机、数据未准备好等因素超过了重试次数从而数据获取失败,会造成数据的缺失以及后期排查难度大等问题。

[0004] 因此,亟需提供一种新的数据请求方法,以解决上述现有技术中所存在的问题。

发明内容

[0005] 本发明的目的在于提供一种请求效率高且不易丢失数据的数据请求方法、电子设备及可读存储介质。

[0006] 为实现上述目的,本发明提供一种数据请求方法,包括步骤:

[0007] S1,发起网络请求,以调用接口获取数据;

[0008] S2,获取接口调用结果,若成功调用,进入步骤S5,否则,进入步骤S3;

[0009] S3,确定所述网络请求对应的异常类型,若所述异常类型为I/O异常,采用HttpClient重试机制发起重试请求,若所述异常类型为业务逻辑异常,采用Spring Retry重试机制发起重试请求,经所述HttpClient重试和/或Spring Retry重试没有成功调用且已达到预先定义的重试阈值,或,所述异常类型为I/O异常、业务逻辑异常之外的其它异常,进入步骤S4;

[0010] S4,将所述网络请求包装成对象并存储;

[0011] S5,结束。

[0012] 较佳地,所述重试阈值包括第一重试阈值和第二重试阈值,在步骤S3中,首先判断所述异常类型是否为I/O异常,若是I/O异常,采用所述HttpClient重试机制发起重试请求,直至成功调用进入步骤S5或超过所述第一重试阈值;若超过所述第一重试阈值或所述异常类型不是I/O异常,判断所述异常类型是否为业务逻辑异常,若是业务逻辑异常,采用所述Spring Retry重试机制发起重试请求,直至成功调用进入步骤S5或超过所述第二重试阈值;若超过所述第二重试阈值或所述异常类型不是业务逻辑异常,进入步骤S4。

[0013] 具体地,在步骤S1之前,还包括步骤:S0,构造HttpClient,定义HttpRequestRetryHandler接口;以及,定义Spring Retry重试模板,设置对应的重试策略和退避策略。

[0014] 较佳地,以单例模式构造HttpClient并设置超时时间,并利用PoolingHttpClientConnectionManager启用Http连接池实现连接复用。

[0015] 较佳地,开启定时扫描空连接的单线程池任务,关闭异常或空闲连接。

[0016] 较佳地,在JVM关闭时,执行addShutdownHook指令关闭HttpClient对象和关闭用于扫描空连接的单线程池。

[0017] 较佳地,在步骤S4之后步骤S5之前,还包括:基于Java反射机制重新执行所述网络请求。

[0018] 为实现上述目的,本发明提供一种电子设备,包括处理器、存储器以及存储在所述存储器中且被配置为由所述处理器执行的计算机程序,所述处理器执行所述计算机程序时,执行如上所述的数据请求方法。

[0019] 为实现上述目的,本发明提供一种计算机可读存储介质,所述计算机可读存储介质存储有计算机程序,所述计算机程序可被处理器执行以完成如上所述的数据请求方法。

[0020] 与现有技术相比,本发明针对异常处理方面,划分为I/O异常、业务逻辑异常、以及I/O异常、业务逻辑异常之外的其它异常(非程序性异常)等多层面维度,对于业务逻辑异常,用Spring Retry处理,而对于I/O异常这种底层异常,则是采用HttpClient重试机制处理,利用HttpRequestRetryHandler异常重试接口能够更底层、更高效解决网络层面异常的问题,减小了程序损耗。且,在经HttpClient重试、Spring Retry重试没有实现成功调用以及非程序性异常情形,则是将网络请求包装成对象存储,以便下次重新触发完成网络请求,避免数据的丢失。

附图说明

[0021] 图1是本发明一实施例数据请求方法的流程示意图。

[0022] 图2是本发明一实施例电子设备的组成结构框图。

具体实施方式

[0023] 为详细说明本发明的内容、构造特征、所实现目的及效果,下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清楚、完整地描述,显然,所描述的实施例仅是本发明一部分实施例,而不是全部的实施例。基于本发明中的实施例,本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例,都属于本发明保护的范围。

[0024] 本发明所提供的数据请求方法包括步骤:

[0025] S0,构造HttpClient,定义HttpRequestRetryHandler接口;以及,定义Spring Retry重试模板,设置对应的重试策略和退避策略。其中,如何构造HttpClient并实现HttpRequestRetryHandler接口,以及Spring Retry的定义和设置为通过现有技术实现。

[0026] S1,发起网络请求,以调用接口获取数据。

[0027] S2,获取接口调用结果,若成功调用,进入步骤S5,否则,进入步骤S3。

[0028] S3,确定网络请求对应的异常类型,若异常类型为I/O异常(IOException异常,发

生I/O错误时引发的异常),采用HttpClient重试机制发起重试请求,若异常类型为业务逻辑异常(不符合业务规则的异常),采用Spring Retry重试机制发起重试请求,经HttpClient重试和/或Spring Retry重试没有成功调用且已达到预先定义的重试阈值,或,异常类型为I/O异常、业务逻辑异常之外的其它异常,进入步骤S4。

[0029] S4,将网络请求包装成对象并存储到数据库或消息队列等存储介质中。

[0030] S5,结束。

[0031] 其中,如何确定网络请求对应的异常类型为现有技术,可以通过例如instanceof运算符实现,在此不再赘述。

[0032] 下面,以一具体实施例为例并结合附图图1对本发明的数据请求方法进行详细说明。

[0033] 请参阅图1,图1示出了本发明数据请求方法的一实施例。如图1所示,在步骤S3中,首先判断异常类型是否为I/O异常,若是I/O异常,则采用HttpClient重试机制发起重试请求,每重试一次后,再次确定是否成功调用,若成功调用,则可以直接进入步骤S5结束当前程序;若没有成功调用,则进一步判断是否超过预先定义的第一重试阈值(重试次数),若没有超过第一重试阈值,则再次采用HttpClient重试机制发起重试请求,直至成功调用进入步骤S5或者超过第一重试阈值。若异常类型不是I/O异常或者超过前述第一重试阈值,进一步判断异常类型是否为业务逻辑异常,若是业务逻辑异常,则采用Spring Retry重试机制发起重试请求,每重试一次后,再次确定是否成功调用,若成功调用,则可以直接进入步骤S5结束当前程序;若没有成功调用,则进一步判断是否超过预先定义的第二重试阈值(重试次数),若没有超过第二重试阈值,则再次采用Spring Retry重试机制发起重试请求,直至成功调用进入步骤S5或者超过第二重试阈值。若异常类型也不是业务逻辑异常或者超过前述第二重试阈值,则直接进入步骤S4进行持久化网络请求(将网络请求包装成对象并存储)。

[0034] 在该实施例中,在步骤S4之后步骤S5之前,还包括:基于Java反射机制重新执行网络请求。由于步骤S4中将网络请求持久化存储,因此,在排查异常原因以及排除异常之后,可以再次执行网络请求,不丢失任何一条数据。步骤S4中持久化的信息包括有Java反射所需的基本信息,如参数、参数类型、方法名、类名等,此外,还包括网络请求的状态和异常信息,以使后期可以获知网络请求是否执行成功以及执行失败的原因,便于后期排查。

[0035] 进一步地,在执行数据请求方法的过程中,还开启定时扫描空连接的单线程池任务,定时周期扫描异常或空闲连接,以清理异常或空闲连接,腾出更多可用的连接。在JVM(Java Virtual Machine,Java虚拟机)关闭时,执行addShutdownHook指令(增加一个关闭钩子)关闭HttpClient对象和关闭用于扫描空连接的单线程池的方法,以实现内存清理、资源回收等,腾出空间。

[0036] 在该实施例中,使用HttpClient发送post和get请求时,需要实例化HttpClient实例,再调用HttpClient.execute()方法,每次实例化HttpClient耗时较大,而且HttpClient实例不能共用,不利于大量的请求处理,考虑到HttpClient实例公用,且HttpClient类线程安全,在该实施例中,以单例模式构造HttpClient。由于Http连接需要三次握手,开销很大,在该实施例中,在构造HttpClient时设置超时时间,利用PoolingHttpClientConnectionManager启用Http连接池实现连接复用,减少创建TCP连接的开销提升性

能,进一步提高网络请求的效率。与此同时,在每次取重用的连接时禁止连接活性检测,以及保持Keep-Alive模式,同样减少了创建TCP连接的开销提升性能,提高网络请求的效率。

[0037] 接下来请参阅图2,本发明所提供的电子设备100包括处理器10、存储器20以及计算机程序30。存储器20存储有一个或多个计算机程序30,例如数据请求程序,当一个或多个计算机程序30被处理器10执行,使得处理器10实现上述数据请求方法。

[0038] 存储器20可以是任何形态的集成在电子设备中的随机存取存储器、只读存储器、快闪存储器或其他类似装置。处理器10可以是中央处理单元或者是其它可程序化的一般用途或者特殊用途的微处理器、数字信号处理器、可程序化控制器、特殊集成电路或其它类似装置或者这些装置的组合。

[0039] 相应地,本发明还涉及一种计算机可读存储介质,计算机可读存储介质存储有计算机程序,计算机程序被处理器10执行时,完成上述实施例中的数据请求方法。其中,计算机程序包括计算机程序代码,计算机程序代码可以为源代码形式、对象代码形式、可执行文件或某些中间形式等。计算机可读存储介质可以包括:能够携带计算机程序代码的任何实体或装置、记录介质、U盘、移动硬盘、磁碟、光盘、计算机存储器、只读存储器(ROM)、随机存取存储器(RAM)等。

[0040] 综上,本发明针对异常处理方面,划分为I/O异常、业务逻辑异常、以及I/O异常、业务逻辑异常之外的其它异常(非程序性异常)等多层面维度,对于业务逻辑异常,用Spring Retry处理,而对于I/O异常这种底层异常,则是采用HttpClient重试机制处理,利用HttpRequestRetryHandler异常重试接口能够更底层、更高效解决网络层面异常的问题,减小了程序损耗。且,在经HttpClient重试、Spring Retry重试没有实现成功调用以及非程序性异常情形,则是将网络请求包装成对象存储,以便下次重新触发完成网络请求,避免数据的丢失。即是,以尽可能小的损耗发起重试以及兜底,保证每一个网络请求都能有迹可循的执行完毕,同时保证效率。

[0041] 以上所揭露的仅为本发明的较佳实例而已,不能以此来限定本发明之权利范围,因此依本发明权利要求所作的等同变化,均属于本发明所涵盖的范围。

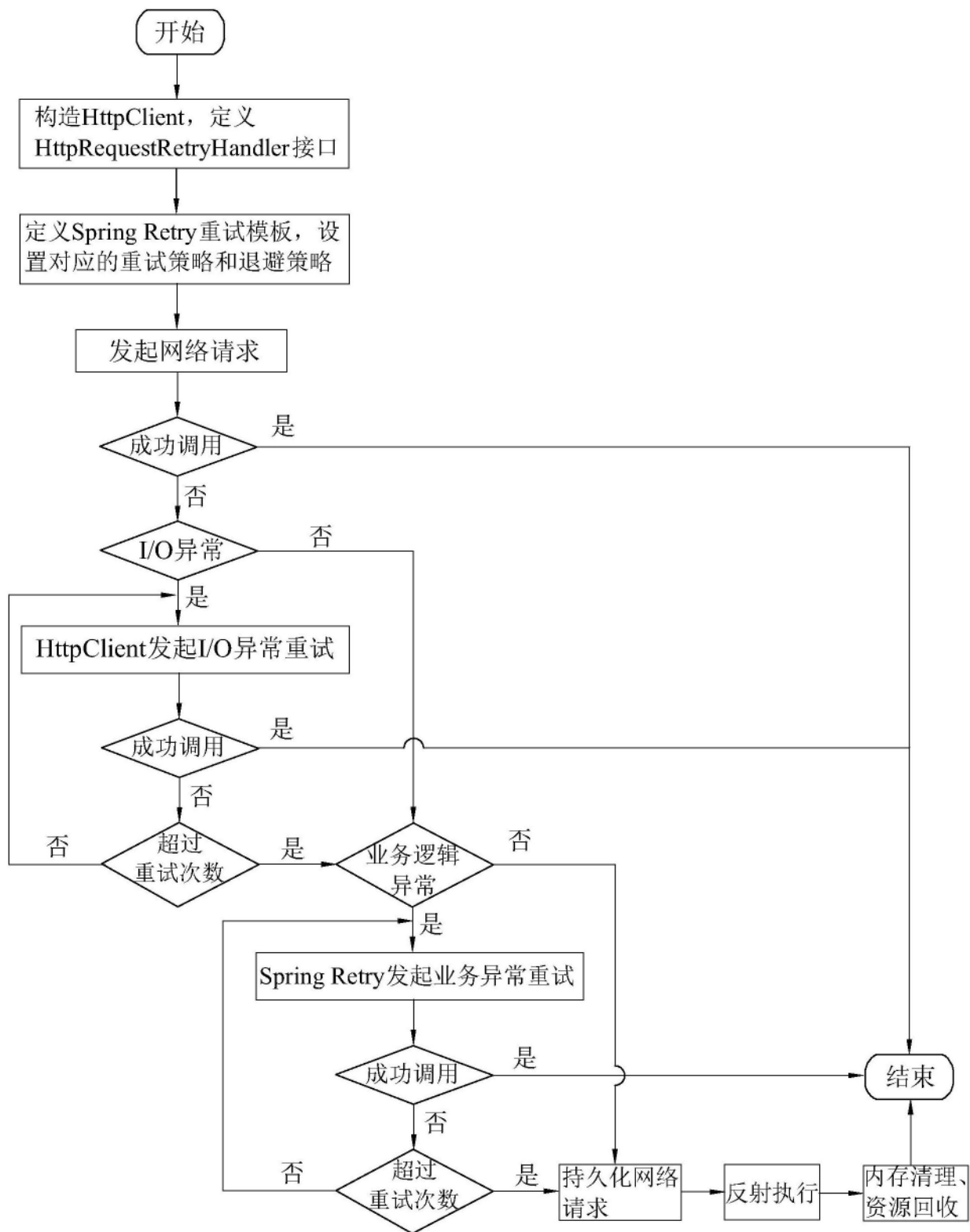


图1

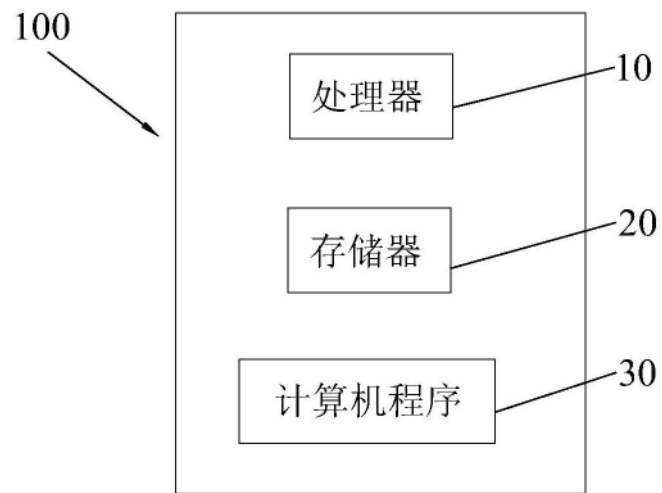


图2