



(21) 申请号 202410336645.6

(22) 申请日 2024.03.22

(71) 申请人 度小满科技(北京)有限公司

地址 100085 北京市海淀区西北旺东路10  
号院西区4号楼度小满金融总部

(72) 发明人 王瀚祺 马涛 徐健 赵辉 王京  
李晓亮

(74) 专利代理机构 北京北汇律师事务所 11711  
专利代理师 盛东生

(51) Int. Cl.

G06F 9/50 (2006.01)

G06F 9/48 (2006.01)

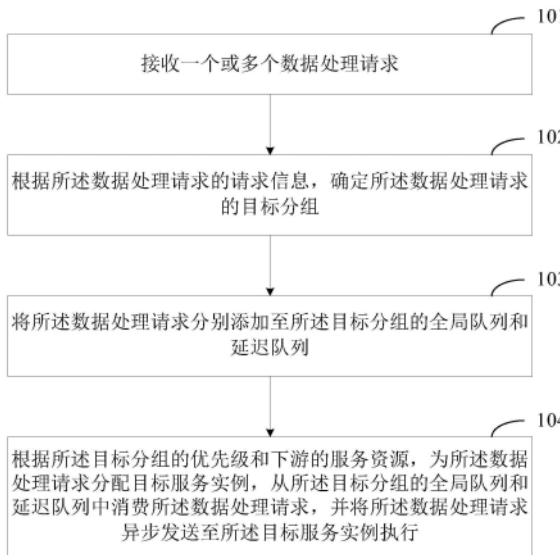
权利要求书2页 说明书12页 附图5页

(54) 发明名称

一种数据处理请求的处理方法和装置

(57) 摘要

本公开实施例提供一种数据处理请求的处理方法和装置,涉及数据处理技术领域。该方法的具体实施方式包括:接收一个或多个数据处理请求;根据数据处理请求的请求信息,确定数据处理请求的目标分组;将数据处理请求分别添加至目标分组的全局队列和延迟队列;根据目标分组的优先级和下游的服务资源,为数据处理请求分配目标服务实例,从目标分组的全局队列和延迟队列中消费数据处理请求,并将数据处理请求异步发送至目标服务实例执行。该实施方式能够确保服务资源的高效利用、避免有限资源被长时间占用,灵活地管理和分配连接资源,缩短请求响应时间、提高处理速度和响应性能,从而提升系统响应效率、运行稳定性和用户体验。



1. 一种数据处理请求的处理方法,其特征在于,包括:  
接收一个或多个数据处理请求;  
根据所述数据处理请求的请求信息,确定所述数据处理请求的目标分组;  
将所述数据处理请求分别添加至所述目标分组的全局队列和延迟队列;  
根据所述目标分组的优先级和下游的服务资源,为所述数据处理请求分配目标服务实例,从所述目标分组的全局队列和延迟队列中消费所述数据处理请求,并将所述数据处理请求异步发送至所述目标服务实例执行。
2. 如权利要求1所述的处理方法,其特征在于,所述根据所述数据处理请求的请求信息,确定所述数据处理请求的目标分组,包括:  
根据请求方类型、请求方备注、来源业务线、和/或请求紧急度中的至少一者,确定所述数据处理请求的请求等级;  
从预构建的等级分组映射关系中,确定与所述请求等级对应的目标分组。
3. 如权利要求2所述的处理方法,其特征在于,所述根据请求方类型、请求方备注、来源业务线、和/或请求紧急度中的至少一者,确定所述数据处理请求的请求等级,包括:  
分别将所述数据处理请求的请求方类型、请求方备注、来源业务线、请求紧急度中的多者的属性值编码,得到多个属性值的编码值;  
将多个编码值进行加权计算,得到所述数据处理请求的加权得分;  
根据加权得分的分值区间与请求等级的映射关系,确定所述数据处理请求的请求等级。
4. 如权利要求1所述的处理方法,其特征在于,所述根据所述目标分组的优先级和下游的服务资源,为所述数据处理请求分配目标服务实例,包括:  
判断所述目标分组是否为高优先级;  
在所述目标分组为高优先级的情况下,判断所述数据处理请求的请求标识是否存在于所述延迟队列;  
针对所述目标分组为高优先级、且请求标识存在于所述延迟队列的数据处理请求,根据所述数据处理请求所需的处理资源、下游的服务资源中各个服务实例的实例容量、和/或实例性能等级,为所述数据处理请求匹配一个或多个目标服务实例。
5. 如权利要求4所述的处理方法,其特征在于,在所述目标分组非高优先级的情况下,根据所述目标分组的优先级和各个分组的请求数阈值,确定所述数据处理请求的响应方式。
6. 如权利要求5所述的处理方法,其特征在于,所述根据所述目标分组的优先级和各个分组的请求数阈值,确定所述数据处理请求的响应方式,包括:  
步骤S1:获取各个分组的全局队列中的数据请求的全局请求数,计算全部分组的全局请求数之和;  
步骤S2:判断所述全局请求数之和是否超过所述目标分组对应的请求数阈值;  
步骤S3:在所述全局请求数之和超过所述目标分组对应的请求数阈值的情况下,暂停所述数据处理请求的出队;  
步骤S4:等待预设时间间隔,转至步骤S1。
7. 如权利要求6所述的处理方法,其特征在于,所述请求数阈值包括第一阈值、第二阈

值和第三阈值;所述判断所述全局请求数之和是否超过所述目标分组对应的请求数阈值,包括:

在所述目标分组为次高优先级且所述全局请求数之和超过所述第一阈值、或者所述目标分组为中优先级且所述全局请求数之和超过所述第二阈值、或者所述目标分组为低优先级且所述全局请求数之和超过所述第三阈值的情况下,确定所述全局请求数之和超过所述目标分组对应的请求数阈值。

8.如权利要求6所述的处理方法,其特征在于,还包括:

在所述全局请求数之和未超过所述目标分组对应的请求数阈值的情况下,判断所述数据处理请求的请求标识是否存在于所述延迟队列;

在所述数据处理请求的请求标识未存在于所述延迟队列的情况下,拒绝所述数据处理请求。

9.一种数据处理请求的处理装置,其特征在于,包括:

接收模块,用于接收一个或多个数据处理请求;

分组模块,用于根据所述数据处理请求的请求信息,确定所述数据处理请求的目标分组;

队列模块,用于将所述数据处理请求分别添加至所述目标分组的全局队列和延迟队列;

消费模块,用于根据所述目标分组的优先级和下游的服务资源,为所述数据处理请求分配目标服务实例,从所述目标分组的全局队列和延迟队列中消费所述数据处理请求,并将所述数据处理请求异步发送至所述目标服务实例执行。

10.一种电子设备,包括:

处理器;以及

存储程序的存储器,

其中,所述程序包括指令,所述指令在由所述处理器执行时使所述处理器执行根据权利要求1-8中任一项所述的数据处理请求的处理方法。

11.一种存储有计算机指令的非瞬时计算机可读存储介质,其中,所述计算机指令用于使所述计算机执行根据权利要求1-8中任一项所述的数据处理请求的处理方法。

## 一种数据处理请求的处理方法和装置

### 技术领域

[0001] 本公开涉及数据处理技术领域,尤其涉及一种数据处理请求的处理方法和装置。

### 背景技术

[0002] 随着计算机技术的快速发展,数据处理请求不断激增,现有的数据处理请求的处理过程中,通常由客户端与网关连接、向网关发起数据处理请求,网关为数据处理请求分配线程,再发送给下游的服务资源。然而,由于线程数有限、各个请求需要的服务资源大小不一,一旦服务资源不足,线程会被请求长时间占用,无法连接其它的紧急请求,导致连接资源利用率低下,用户长时间等待却未得到响应,使用体验较差、投诉和流失严重。

[0003] 为了保证数据处理请求的及时响应,通常会引入消息队列对数据处理请求进行整理,再分配至服务实例执行,常用的消息队列工具包括Kafka、Beanstalkd、Disruptor等。

[0004] 然而,Kafka通过将消息写入磁盘或者从磁盘读取消息实现持久化,但是Kafka作为消息队列过于庞大,架构逻辑复杂,每个消息的队列进出耗时较长,导致请求处理速度较慢、性能低下,需要消耗的计算资源较高;数据处理请求的类型正在朝向多样化发展,导致各式数据处理请求的处理耗时长短不一,Beanstalkd并不存在最大内存控制用以限制服务实例的大小,导致耗时的数据处理请求长时间占用多个服务实例,无法及时释放可用资源,服务稳定性差;Disruptor仅能适用于消息量较少的场景,无法处理海量的数据处理请求。

### 发明内容

[0005] 有鉴于此,本公开实施例提供一种数据处理请求的处理方法和装置,能够解决连接资源未合理分配、有限资源被长时间占用,请求响应时间过长、处理速度慢且性能低下,进而导致系统响应效率和运行稳定性较差的问题。

[0006] 为实现上述目的,根据本公开的一方面,提供了一种数据处理请求的处理方法,包括:

[0007] 接收一个或多个数据处理请求;

[0008] 根据所述数据处理请求的请求信息,确定所述数据处理请求的目标分组;

[0009] 将所述数据处理请求分别添加至所述目标分组的全局队列和延迟队列;

[0010] 根据所述目标分组的优先级和下游的服务资源,为所述数据处理请求分配目标服务实例,从所述目标分组的全局队列和延迟队列中消费所述数据处理请求,并将所述数据处理请求异步发送至所述目标服务实例执行。

[0011] 根据本公开的另一方面,提供了一种数据处理请求的处理装置,包括:

[0012] 接收模块,用于接收一个或多个数据处理请求;

[0013] 分组模块,用于根据所述数据处理请求的请求信息,确定所述数据处理请求的目标分组;

[0014] 队列模块,用于将所述数据处理请求分别添加至所述目标分组的全局队列和延迟

队列；

[0015] 消费模块,用于根据所述目标分组的优先级和下游的服务资源,为所述数据处理请求分配目标服务实例,从所述目标分组的全局队列和延迟队列中消费所述数据处理请求,并将所述数据处理请求异步发送至所述目标服务实例执行。

[0016] 根据本公开的再一方面,提供了一种电子设备,包括:

[0017] 处理器;以及

[0018] 存储程序的存储器,

[0019] 其中,所述程序包括指令,所述指令在由所述处理器执行时使所述处理器执行所述数据处理请求的处理方法。

[0020] 根据本公开实施例的还一个方面,提供了一种存储有计算机指令的非瞬时计算机可读存储介质,其中,所述计算机指令用于使所述计算机执行所述数据处理请求的处理方法。

[0021] 本申请实施例中提供的一个或多个技术方案,通过引入不同优先级的分组的全局队列和延迟队列进行缓冲,可以实现从接收请求直至发送至目标实例的整个处理流程皆采用全程异步非阻塞的方式,避免了下游处理耗时不稳定引发的整体效率和稳定性降低,极大地减少了不必要的等待时间,充分利用计算资源,使系统得以稳定运行,提高用户的使用体验;并且,对于不同分组采用限流逻辑,不立即拒绝访问,在请求超时之前智能决策,充分考虑服务资源、请求成功率和请求性能,智能地应对请求流量的涨跌,提高系统的鲁棒性和可用性,实现请求的毫秒级流量缓冲机制,相较于现有的队列工具大大缩短队列耗时的技术效果。

## 附图说明

[0022] 在下面结合附图对于示例性实施例的描述中,本公开的更多细节、特征和优点被公开,在附图中:

[0023] 图1示出了根据本公开示例性实施例的数据处理请求的处理方法的流程图;

[0024] 图2示出了根据本公开示例性实施例的数据处理请求的请求分组的确定方法的流程图;

[0025] 图3示出了根据本公开示例性实施例的数据处理请求的请求等级的确定方法的流程图;

[0026] 图4示出了根据本公开示例性实施例的数据处理请求的限流方法的流程图;

[0027] 图5示出了根据本公开示例性实施例的请求数阈值的判断方法的流程图;

[0028] 图6示出了根据本公开示例性实施例的数据处理请求的处理装置的示意性框图;

[0029] 图7示出了能够用于实现本公开的实施例的示例性电子设备的结构框图。

## 具体实施方式

[0030] 下面将参照附图更详细地描述本公开的实施例。虽然附图中显示了本公开的某些实施例,然而应当理解的是,本公开可以通过各种形式来实现,而且不应该被解释为限于这里阐述的实施例,相反提供这些实施例是为了更加透彻和完整地理解本公开。应当理解的是,本公开的附图及实施例仅用于示例性作用,并非用于限制本公开的保护范围。

[0031] 应当理解,本公开的方法实施方式中记载的各个步骤可以按照不同的顺序执行,和/或并行执行。此外,方法实施方式可以包括附加的步骤和/或省略执行示出的步骤。本公开的范围在此方面不受限制。

[0032] 本文使用的术语“包括”及其变形是开放性包括,即“包括但不限于”。术语“基于”是“至少部分地基于”。术语“在本公开实施例中”表示“至少一个实施例”;术语“另一示例性实施例”表示“至少一个另外的实施例”。其他术语的相关定义将在下文描述中给出。需要注意,本公开中提及的“第一”、“第二”等概念仅用于对不同的装置、模块或单元进行区分,并非用于限定这些装置、模块或单元所执行的功能的顺序或者相互依存关系。

[0033] 需要注意,本公开中提及的“一个”、“多个”的修饰是示意性而非限制性的,本领域技术人员应当理解,除非在上下文另有明确指出,否则应该理解为“一个或多个”。

[0034] 本公开实施方式中的多个装置之间所交互的消息或者信息的名称仅用于说明性的目的,而并不是用于对这些消息或信息的范围进行限制。

[0035] **Kafka**: 一个分布式流数据平台,旨在处理和传输大规模的实时数据流,以及在分布式环境下构建高可用性、高吞吐量的数据管道。最初由LinkedIn公司开发,后成为Apache软件基金会的开源项目。Kafka通过分布式发布-订阅系统以及磁盘顺序写入的方式实现了高吞吐量和持久性存储的能力,可以在包括消息分发、日志处理、数据流处理等方面发挥重要作用,是目前被应用最为广泛的消息队列之一。

[0036] **Beanstalkd**: 一个简单快速的分布式工作队列系统,协议基于ASCII编码运行在TCP上。Beanstalkd最初设计的目的是通过后台异步执行耗时任务的方式降低高容量Web应用的页面延时,相较于Kafka等消息队列,具有简单、轻量、易用等特点,同时还有众多语言版本的客户端支持,以上优点使得Beanstalkd成为各种需要消息队列系统场景的常见选择。

[0037] **Disruptor**: 是英国外汇交易公司LMAX开发的一个高性能队列,研发的初衷是解决内存队列的延迟问题。基于Disruptor开发的系统单线程能支撑每秒上百万订单,被设计用于在生产者-消费者(producer-consumer problem,简称PCP)问题上获得尽量高的吞吐量(TPS)和尽量低的延迟。

[0038] **Netty**: 是由JBoss(基于J2EE<Java 2Platform Enterprise Edition,用于应用程序开发的标准平台>的开放源代码的应用服务器)提供的一个Java开源框架,用以提供高性能、高可靠性的、异步的、基于事件驱动的网络应用程序的开发。Netty是一个基于NIO(non-blocking IO,即非阻塞IO)的网络编程框架,也即,Netty中的所有IO(Input/Output,即输入/输出)操作皆为异步、非阻塞的。Netty包括Reactor通信调度层、职责链Pipeline层和业务逻辑处理层,Reactor用于监听网络中的连接和读写操作,负责将网络层的数据读取到内存缓冲区中,之后将各种网络事件(比如,连接创建、连接激活、读/写事件等)触发至Pipeline中;Pipeline负责各种事件在职责链中有序地向前(后)传播,选择监听和处理自己关心的事件,动态编排职责链;业务逻辑处理层包括应用层协议管理和业务逻辑处理两类,用于实际响应职责链中的各种事件。

[0039] 以下参照附图描述本公开的方案。

[0040] 图1示出了根据本公开示例性实施例的数据处理请求的处理方法的流程图,如图1所示,本公开的数据处理请求的处理方法包括如下步骤:

[0041] 步骤101,接收一个或多个数据处理请求。

[0042] 在本公开实施例中,数据处理请求的请求方类型、来源业务线、请求紧急度等多种多样,比如,请求方类型可以是用户或者内部人员,来源业务线可能是保险、理财、信贷、支付等,请求紧急度可能是紧急、一般、迟缓等。并且,各种数据处理请求所需的处理时间、处理资源等也各不相同,比如,数据读取请求所需的处理时间较短、数据分析请求所需的处理资源较多等。

[0043] 步骤102,根据所述数据处理请求的请求信息,确定所述数据处理请求的目标分组。

[0044] 相较于现有的请求处理过程中,各种队列工具仅仅按照请求时间排序处理、或者服务器资源达到峰值后直接全部拒绝等,在本公开实施例中,根据数据处理请求的请求等级,将数据处理请求分组,加入不同等级分组中的队列,可以保证等级高及时响应和执行、避免阻塞影响用户使用体验,失效请求及时清理、避免资源的过度占用,以为其它有效请求的处理释放计算资源、保证有效请求的及时处理。

[0045] 进一步地,请求信息包括请求标识、请求方类型、请求方备注、来源业务线、请求紧急度、所需处理资源、过期时间等,如图2所示,本公开的数据处理请求的请求分组的确定方法包括如下步骤:

[0046] 步骤201,根据请求方类型、请求方备注、来源业务线、和/或请求紧急度中的至少一者,确定所述数据处理请求的请求等级。

[0047] 在本公开实施例中,对于请求方类型,由于内部人员使用数据通常用于数据分析、数据测试等,数据分析的分析类型根据分析目的各有不同,包括用户分析、异常分析、趋势分析、单点分析、综合分析等,故而内部人员的数据使用通常非紧急情况,可以安排在空闲时间段,因此通常情况下,请求方类型为用户的数据处理请求的请求等级高于请求方类型为内部人员的数据处理请求;

[0048] 对于请求方备注,可能存在数据分析后对用户进行质量调整,用户可能由低质量变为高质量(即正向调整)、或者由高质量变为低质量(即负向调整),不同的质量调整方式对应不同的请求方备注、不同的请求方备注对应不同的请求等级的调整方式,比如,请求方备注为正向调整时提高数据处理请求的请求等级,请求方备注为负向调整时降低数据处理请求的请求等级;

[0049] 对于来源业务线,可以根据来源业务线的业务线类型进行选择设置,比如,业务线类型为保险的数据处理请求的请求等级低于业务线类型为理财和支付的数据处理请求、业务线类型为理财和支付的数据处理请求的请求等级低于业务线类型为信贷的数据处理请求;

[0050] 对于请求紧急度,可以根据数据处理请求是否需要即时反馈数据处理结果确定,比如,需要即时反馈数据处理结果的数据处理请求的请求紧急度高于无需即时反馈数据处理结果的数据处理请求,并且,需要即时反馈的反馈时间越短,请求紧急度越高,相应地,请求紧急度越高,数据处理请求的请求等级越高。

[0051] 进一步地,数据处理请求的请求等级的确定可以根据实际的数据处理场景进行选择设置,通过请求方类型、请求方备注、来源业务线、请求紧急度中的至少一者确定,比如,请求紧急度越高,数据处理请求的请求等级越高、。

[0052] 更进一步地,如图3所示,在数据处理请求的请求等级通过请求方类型、请求方备注、来源业务线、请求紧急度中的多者确定的情况下,本公开的数据处理请求的请求等级的确定方法包括如下步骤:

[0053] 步骤301,分别将所述数据处理请求的请求方类型、请求方备注、来源业务线、请求紧急度中的多者的属性值编码,得到多个属性值的编码值。

[0054] 在本公开实施例中,根据请求方类型、请求方备注、来源业务线、请求紧急度对请求等级的影响程度,对请求方类型、请求方备注、来源业务线、请求紧急度的属性值按照百分制编码,其中,影响程度越大、编码值越大。比如,请求方类型包括用户和内部人员,用户对请求等级的影响程度大、属性值编码为70分,内部人员对请求等级的影响程度小、属性值编码为30分;请求方备注包括正向调整和负向调整,正向调整对请求等级的影响为提高等级、属性值编码为200分,内部人员对请求等级的影响为降低等级、属性值编码为-100分;来源业务线包括保险、理财、信贷、支付、搜索等,信贷、搜索、理财和支付、保险对请求等级的影响程度顺次减小、属性值顺次编码为40分、30分、20分、10分;请求紧急度包括紧急、一般、迟缓,属性值顺次编码为60分、30分、10分。

[0055] 进一步地,在数据处理请求的请求等级通过请求方类型、请求方备注、来源业务线、请求紧急度中的多者确定的情况下,无需对全部项的请求信息编码,仅对确定数据处理请求的请求等级的各项请求信息编码即可。比如,在请求等级由请求方类型、请求方备注确定的情况下,对请求方类型、请求方备注的属性值编码即可;在请求等级由请求方类型、来源业务线确定的情况下,对请求方类型、来源业务线的属性值编码即可;在请求等级由请求方类型、请求紧急度确定的情况下,对请求方类型、请求紧急度的属性值编码即可;在请求等级由请求方备注、来源业务线确定的情况下,对请求方备注、来源业务线的属性值编码即可;在请求等级由请求方备注、请求紧急度确定的情况下,对请求方备注、请求紧急度的属性值编码即可;在请求等级由来源业务线、请求紧急度确定的情况下,对来源业务线、请求紧急度的属性值编码即可;在请求等级由请求方类型、请求方备注、来源业务线确定的情况下,对请求方类型、请求方备注、来源业务线的属性值编码即可;在请求等级由请求方类型、请求方备注、请求紧急度确定的情况下,对请求方类型、请求方备注、请求紧急度的属性值编码即可;在请求等级由请求方备注、来源业务线、请求紧急度确定的情况下,对请求方备注、来源业务线、请求紧急度的属性值编码即可;在请求等级由请求方类型、来源业务线、请求紧急度确定的情况下,对请求方类型、来源业务线、请求紧急度的属性值编码即可;在请求等级由请求方类型、请求方备注、来源业务线、请求紧急度确定的情况下,对请求方类型、请求方备注、来源业务线、请求紧急度的属性值编码。

[0056] 步骤302,将多个编码值进行加权计算,得到所述数据处理请求的加权得分。

[0057] 在本公开实施例中,将请求方类型、请求方备注、来源业务线、请求紧急度的属性值按照百分制编码,根据编码值对应的请求信息的加权权重,对多个编码值加权计算,得到数据处理请求的加权得分。

[0058] 进一步地,多个编码值对应的请求信息的加权权重之和为1,各项请求信息的加权权重可以相等或者不等,根据实际的数据处理场景进行选择设置即可,比如,请求方类型的加权权重为0.3、请求方备注的加权权重为0.2、来源业务线的加权权重为0.1、请求紧急度的加权权重为0.4。



[0059] 更进一步地,在数据处理请求的请求等级通过请求方类型、请求方备注、来源业务线、请求紧急度中的多者确定的情况下,仅确定数据处理请求的请求等级的各项请求信息的加权重之和为1。其中,确定数据处理请求的请求等级的各项请求信息的加权重可以相等或者不等,根据实际的数据处理场景进行选择设置即可,比如,在请求等级由请求方类型、请求方备注确定的情况下,请求方类型与请求方备注两者的加权重之和为1;在请求等级由请求方类型、请求方备注、来源业务线确定的情况下,请求方类型、请求方备注、来源业务线三者的加权重之和为1。

[0060] 步骤303,根据加权得分的分值区间与请求等级的映射关系,确定所述数据处理请求的请求等级。

[0061] 在本公开实施例中,预先设置加权得分的分值区间与请求等级的映射关系,以根据数据处理请求的加权得分确定数据处理请求的请求等级,比如,加权得分的分值区间与请求等级的映射关系为:加权得分[75,100]的数据处理请求的请求等级为高级、加权得分[55,75)的数据处理请求的请求等级为次高级、加权得分[40,55)的数据处理请求的请求等级为中级、加权得分[0,40]的数据处理请求的请求等级为低级。

[0062] 在本公开实施例中,通过本公开的数据处理请求的请求等级的确定方法,面对数据处理请求的请求等级由多项请求信息确定的场景,对多项请求信息的属性值编码、并对编码进行加权计算后确定数据处理请求的加权得分,进而根据加权得分所属的分值区间确定数据处理请求的请求等级,后续即可将数据处理请求分配至对应请求等级的分组内,以便于优先处理请求等级高的数据处理请求,提升用户服务使用体验,避免有限资源被长时间占用,提高请求处理速度,提升系统响应效率和运行稳定性。

[0063] 步骤202,从预构建的等级分组映射关系中,确定与所述请求等级对应的目标分组。

[0064] 在本公开实施例中,等级分组映射关系中存储了各个请求等级与各个分组的优先级的对应关系,分组的优先级的等级个数与请求等级的等级个数相同,确定数据处理请求的请求等级即可确定数据处理请求的分组优先级,进而确定数据处理请求所属的目标分组。比如,等级分组映射关系中请求等级的属性值为高级、次高级、中级、低级,对应的分组的优先级分别为高优先级、次高优先级、中优先级、低优先级,在数据处理请求的请求等级为中级的情况下,目标分组即为中优先级的分组。

[0065] 进一步地,优先级越高,分组中的数据处理请求越会被优先分配服务资源以执行数据处理请求。

[0066] 在本公开实施例中,通过本公开的数据处理请求的请求分组的确定方法,能够根据数据处理请求的请求等级将全部数据处理请求分组,后续即可根据分组采取不同的应对措施,提升用户服务使用体验,防止单个请求长时间占用资源而其他请求无法得到处理,提高请求处理速度,提升系统响应效率和运行稳定性。

[0067] 步骤103,将所述数据处理请求分别添加至所述目标分组的全局队列和延迟队列。

[0068] 在本公开实施例中,每个分组包括一个全局队列和一个延迟队列,全局队列存储了被分配至该分组的数据处理请求的请求标识、待处理数据和处理方式等,以对同一请求等级的数据处理请求排序,从而将同一请求等级的数据处理请求根据入队时间按序出队处理;延迟队列存储了被分配至该分组的各个数据处理请求的请求标识和过期时间,用于

判断数据处理请求是否过期,各个数据处理请求的过期时间按照由远及近排序。两个队列分开设计,以确保数据处理请求在其有效时间内被且仅被处理一次,防止过期的请求被处理。

[0069] 进一步地,将数据处理请求的待处理数据和处理方式添加至目标分组的全局队列、并将数据处理请求的过期时间添加值目标分组的延迟队列,从而后续可以根据分组的优先级以及是否过期为数据处理请求分配服务资源,保证高优先级的数据处理请求立即响应、其它优先级的数据处理请求分流等待处理,提升用户使用体验,避免有限资源被长时间占用处理无需紧急处理的请求,提升请求处理速度,进而提高系统响应效率和运行稳定性。

[0070] 更进一步地,延迟队列会实时判断当前时间下是否存在过期的数据处理请求,在数据处理请求过期时及时消费,避免有限资源被过期请求长时间占用,浪费系统资源;全局队列会在队列为空时进入等待状态,避免资源消耗。

[0071] 步骤104,根据所述目标分组的优先级和下游的服务资源,为所述数据处理请求分配目标服务实例,从所述目标分组的全局队列和延迟队列中消费所述数据处理请求,并将所述数据处理请求异步发送至所述目标服务实例执行。

[0072] 在本公开实施例中,下游的服务资源包括下游各个服务实例的实例容量、实例性能等级等实例信息,实例容量表征了服务实例可用的计算资源的大小,由于同样的实例容量下,性能越好的服务实例处理数据处理请求的速度越快,因此,利用实例性能等级表征服务实例的性能优良,从而在分配服务资源时,可以为优先级高的分组匹配实例性能等级高的服务实例,以提升请求等级较高的数据处理请求的处理效率,及时响应用户,提升用户使用体验。

[0073] 进一步地,在各个分组的数据处理请求出队时,不同优先级分组的限流逻辑各不相同,其中,不对高优先级的分组限流,只要下游的服务资源正常供给,即为高优先级的分组中的数据处理请求分配服务实例,也即,只要下游的服务资源正常供给,高优先级的分组中的数据处理请求即正常出列;对于次高优先级、中优先级、低优先级的分组进行请求数限流,不同的优先级的分组对应不同的请求数阈值,超过对应的请求数阈值即暂停对应分组的数据处理请求出列,从而实现不同粒度分组的数据处理请求的不同响应方式,提升用户使用体验。

[0074] 更进一步地,如图4所示,本公开的数据处理请求的限流方法包括如下步骤:

[0075] 步骤401,判断所述目标分组是否为高优先级,如果是,转至步骤402;如果不是,转至步骤405。

[0076] 在本公开实施例中,由于不对高优先级的分组限流,因此,在处理数据处理请求时,首先判断数据处理请求所属的目标分组是否为高优先级,对高优先级分组的数据处理请求优先响应,也即,只要下游的服务资源不崩溃,就会优先服务高优先级分组的数据处理请求。

[0077] 步骤402,判断所述数据处理请求的请求标识是否存在于所述延迟队列,如果是,转至步骤403;如果不是,转至步骤404。

[0078] 在本公开实施例中,在目标分组为高优先级的情况下,再判断数据处理请求是否存在于延迟队列。

[0079] 步骤403,根据所述数据处理请求所需的处理资源、下游的服务资源中各个服务实例的实例容量、和/或实例性能等级,为所述数据处理请求匹配一个或多个目标服务实例。

[0080] 在本公开实施例中,在数据处理请求存在于延迟队列的情况下,表示数据处理请求未过期,故而根据数据处理请求所需的处理资源、下游的服务资源中各个服务实例的实例信息,为数据处理请求匹配处理资源。

[0081] 步骤404,拒绝所述数据处理请求。

[0082] 在本公开实施例中,在数据处理请求未存在于延迟队列的情况下,表示数据处理请求已过期,故而直接拒绝数据处理请求,避免了过期请求长时间占用服务资源现象的发生,防止服务资源的浪费,提升请求处理效率。

[0083] 步骤405,获取各个分组的全局队列中的数据处理请求的全局请求数,计算全部分组的全局请求数之和。

[0084] 在本公开实施例中,在目标分组非高优先级的情况下,目标分组存在对应的限流逻辑,故而需要根据当下负荷的全局请求数,结合目标分组的请求数阈值,确定数据处理请求的响应方式。

[0085] 步骤406,判断所述全局请求数之和是否超过所述目标分组对应的请求数阈值,如果是,转至步骤407;如果否,转至步骤402。

[0086] 在本公开实施例中,请求数阈值包括第一阈值、第二阈值、第三阈值,不同阈值分别对应不同的优先级分组,其中,次高优先级的分组对应第一阈值、中优先级的分组对应第二阈值、低优先级的分组对应第三阈值。其中,第一阈值至第三阈值可以根据实际的处理场景进行选择设置,比如,第一阈值为8000个、第二阈值为6000个、第三阈值为4000个。

[0087] 进一步地,如图5所示,本公开的请求数阈值的判断方法包括如下步骤:

[0088] 步骤501,判断所述目标分组是否为次高优先级,如果是,转至步骤502;如果否,转至步骤503。

[0089] 在本公开实施例中,由于次高优先级、中优先级、低优先级的分组分别存在不同的请求数阈值,因此,需要根据目标分组的优先级,确定目标分组对应的请求数阈值,再与全局请求数对比确定响应方式。

[0090] 进一步地,按照优先级由高到低的顺序,确定目标分组所属的优先级,先判断目标分组是否是次高优先级。

[0091] 步骤502,判断所述全局请求数之和是否超过第一阈值,如果是,转至步骤503,如果否,转至步骤505。

[0092] 在本公开实施例中,在目标分组为次高优先级的情况下,判断当下负荷的全局请求数是否超过第一阈值。

[0093] 步骤503,确定所述全局请求数之和超过所述目标分组对应的请求数阈值。

[0094] 在本公开实施例中,在目标分组为次高优先级且全局请求数之和超过第一阈值、或者目标分组为中优先级且全局请求数之和超过第二阈值、或者目标分组为低优先级且全局请求数之和超过第三阈值的情况下,确定全局请求数之和超过目标分组对应的请求数阈值。

[0095] 步骤504,确定所述全局请求数之和未超过所述目标分组对应的请求数阈值。

[0096] 在本公开实施例中,在目标分组为次高优先级且全局请求数之和未超过第一阈

值、或者目标分组为中优先级且全局请求数之和未超过第二阈值、或者目标分组为低优先级且全局请求数之和未超过第三阈值的情况下,确定全局请求数之和未超过目标分组对应的请求数阈值。

[0097] 步骤505,判断所述目标分组是否为中优先级,如果是,转至步骤506;如果否,转至步骤507。

[0098] 在本公开实施例中,在目标分组非次高优先级的情况下,继续判断目标分组是否为中优先级。

[0099] 步骤506,判断所述全局请求数之和是否超过第二阈值,如果是,转至步骤503,如果否,转至步骤504。

[0100] 在本公开实施例中,在目标分组为中优先级的情况下,判断当下负荷的全局请求数是否超过第二阈值。

[0101] 步骤507,判断所述全局请求数之和是否超过第三阈值,如果是,转至步骤503,如果否,转至步骤504。

[0102] 在本公开实施例中,在目标分组既非次高优先级、也非中优先级的情况下,目标分组为低优先级分组,故而判断当下负荷的全局请求数是否超过第三阈值。

[0103] 在本公开实施例中,通过本公开的请求数阈值的判断方法,对于非高优先级的分组进行限流,确定全局请求数是否超过各个优先级的分组的请求数阈值,进而后续可以确定是对数据处理请求暂停处理还是分配服务资源处理,从而实现数据处理请求的分级处理,保证高优先级的即时响应、避免低优先级的长时间占用服务资源,提升请求处理效率和用户服务使用体验。

[0104] 步骤407,暂停所述数据处理请求的出队。

[0105] 在本公开实施例中,在全局请求数之和超过目标分组对应的请求数阈值的情况下,暂停数据处理请求的出队,等待下游服务资源的释放,防止服务超载导致的崩溃,保证系统的正常响应和稳定运行。

[0106] 步骤408,等待预设时间间隔,转至步骤405。

[0107] 在本公开实施例中,预设时间间隔可以根据实际的数据请求处理场景进行选择性的设置,次高优先级、中优先级、低优先级的分组的预设时间间隔可以相等或者不等,比如,次高优先级的预设时间间隔低于中优先级的预设时间间隔、中优先级的预设时间间隔低于低优先级的预设时间间隔。

[0108] 在本公开实施例中,通过本公开的数据处理请求的限流方法,对高优先级的数据处理请求的出队不限流直接响应、非高优先级的数据处理请求的出队应用限流逻辑,实现不同优先级的数据处理请求的分级响应,从而保证高优先级的即时响应、避免有限资源被优先级偏下等级的数据处理请求长时间占用,提升数据处理请求的处理效率和用户服务使用体验,提高系统响应效率和运行稳定性。

[0109] 在本公开实施例中,匹配目标服务实例后,利用异步消费工具同时从全局队列和延迟队列中消费数据处理请求,并将数据处理请求发送至目标服务实例,由目标服务实例按照数据处理请求的处理方式对数据处理请求中的待处理数据进行处理。其中,异步消费工具可以与各个分组的队列相区分,确保各个队列的处理能力不会受到下游的影响。

[0110] 进一步地,异步消费工具可以是利用Netty的NIO自定义框架开发的应用程序。

[0111] 在本公开实施例中,相较于现有的消息队列工具过于庞大、连接不做限制导致有限资源被长时间占用、无法处理海量请求的缺陷,通过本公开的数据处理请求的处理方法,引入不同优先级的分组队列,适配海量请求的业务场景,利用数据处理请求的分级响应,确保服务资源的高效利用、避免有限资源被长时间占用,灵活地管理和分配连接资源,缩短请求响应时间、提高处理速度和响应性能,保证用户请求能够在合理的时间内得到响应,从而提升系统响应效率、运行稳定性和用户体验,降低系统崩溃风险,助力系统健康运行。

[0112] 图6是根据本公开实施例的数据处理请求的处理装置的主要模块的示意图,如图6所示,本公开的数据处理请求的处理装置600包括:

[0113] 接收模块601,用于接收一个或多个数据处理请求。

[0114] 分组模块602,用于根据所述数据处理请求的请求信息,确定所述数据处理请求的目标分组。

[0115] 队列模块603,用于将所述数据处理请求分别添加至所述目标分组的全局队列和延迟队列。

[0116] 消费模块604,用于根据下游的服务资源,为所述数据处理请求分配目标服务实例,从所述目标分组的全局队列和延迟队列中消费所述数据处理请求,并将所述数据处理请求异步发送至所述目标服务实例执行。

[0117] 本公开示例性实施例还提供一种电子设备,包括:至少一个处理器;以及与至少一个处理器通信连接的存储器。所述存储器存储有能够被所述至少一个处理器执行的计算机程序,所述计算机程序在被所述至少一个处理器执行时用于使所述电子设备执行根据本公开实施例的方法。

[0118] 本公开示例性实施例还提供一种存储有计算机程序的非瞬时计算机可读存储介质,其中,所述计算机程序在被计算机的处理器执行时用于使所述计算机执行根据本公开实施例的方法。

[0119] 本公开示例性实施例还提供一种计算机程序产品,包括计算机程序,其中,所述计算机程序在被计算机的处理器执行时用于使所述计算机执行根据本公开实施例的方法。

[0120] 参考图7,现将描述可以作为本公开的服务器或客户端的电子设备700的结构框图,其是可以应用于本公开的各方面的硬件设备的示例。电子设备旨在表示各种形式的数字电子的计算机设备,诸如,膝上型计算机、台式计算机、工作台、个人数字助理、服务器、刀片式服务器、大型计算机、和其它适合的计算机。电子设备还可以表示各种形式的移动装置,诸如,个人数字处理、蜂窝电话、智能电话、可穿戴设备和其它类似的计算装置。本文所示的部件、它们的连接和关系、以及它们的功能仅仅作为示例,并且不意在限制本文中描述的和/或者要求的本公开的实现。

[0121] 如图7所示,电子设备700包括计算单元701,其可以根据存储在只读存储器(ROM)702中的计算机程序或者从存储单元708加载到随机访问存储器(RAM)703中的计算机程序,来执行各种适当的动作和处理。在RAM703中,还可存储设备700操作所需的各种程序和数据。计算单元701、ROM702以及RAM703通过总线704彼此相连。输入/输出(I/O)接口705也连接至总线704。

[0122] 电子设备700中的多个部件连接至I/O接口705,包括:输入单元706、输出单元707、存储单元708以及通信单元709。输入单元706可以是能向电子设备700输入信息的任何类型

的设备,输入单元706可以接收输入的数字或字符信息,以及产生与电子设备的用户设置和/或功能控制有关的键信号输入。输出单元707可以是能呈现信息的任何类型的设备,并且可以包括但不限于显示器、扬声器、视频/音频输出终端、振动器和/或打印机。存储单元708可以包括但不限于磁盘、光盘。通信单元709允许电子设备700通过诸如因特网的计算机网络和/或各种电信网络与其他设备交换信息/数据,并且可以包括但不限于调制解调器、网卡、红外通信设备、无线通信收发机和/或芯片组,例如蓝牙™设备、WiFi设备、WiMax设备、蜂窝通信设备和/或类似物。

[0123] 计算单元701可以是各种具有处理和计算能力的通用和/或专用处理组件。计算单元701的一些示例包括但不限于中央处理单元(CPU)、图形处理单元(GPU)、各种专用的人工智能(AI)计算芯片、各种运行机器学习模型算法的计算单元、数字信号处理器(DSP)、以及任何适当的处理器、控制器、微控制器等。计算单元701执行上文所描述的各个方法和处理。例如,在一些实施例中,图1至图5的方法可被实现为计算机软件程序,其被有形地包含于机器可读介质,例如存储单元708。在一些实施例中,计算机程序的部分或者全部可以经由ROM702和/或通信单元709而被载入和/或安装到电子设备700上。在一些实施例中,计算单元701可以通过其他任何适当的方式(例如,借助于固件)而被配置为执行图1至图5的方法。

[0124] 用于实施本公开的方法的程序代码可以采用一个或多个编程语言的任何组合来编写。这些程序代码可以提供给通用计算机、专用计算机或其他可编程数据处理装置的处理器或控制器,使得程序代码当由处理器或控制器执行时使流程图和/或框图中所规定的功能/操作被实施。程序代码可以完全在机器上执行、部分地在机器上执行,作为独立软件包部分地在机器上执行且部分地在远程机器上执行或完全在远程机器或服务器上执行。

[0125] 在本公开的上下文中,机器可读介质可以是有形的介质,其可以包含或存储以供指令执行系统、装置或设备使用或与指令执行系统、装置或设备结合地使用的程序。机器可读介质可以是机器可读信号介质或机器可读储存介质。机器可读介质可以包括但不限于电子的、磁性的、光学的、电磁的、红外的、或半导体系统、装置或设备,或者上述内容的任何合适组合。机器可读存储介质的更具体示例会包括基于一个或多个线的电气连接、便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦除可编程只读存储器(EPROM或快闪存储器)、光纤、便捷式紧凑盘只读存储器(CD-ROM)、光学储存设备、磁储存设备、或上述内容的任何合适组合。

[0126] 如本公开使用的,术语“机器可读介质”和“计算机可读介质”指的是用于将机器指令和/或数据提供给可编程处理器的任何计算机程序产品、设备、和/或装置(例如,磁盘、光盘、存储器、可编程逻辑装置(PLD)),包括,接收作为机器可读信号的机器指令的机器可读介质。术语“机器可读信号”指的是用于将机器指令和/或数据提供给可编程处理器的任何信号。

[0127] 为了提供与用户的交互,可以在计算机上实施此处描述的系统和技术,该计算机具有:用于向用户显示信息的显示装置(例如,CRT(阴极射线管)或者LCD(液晶显示器)监视器);以及键盘和指向装置(例如,鼠标或者轨迹球),用户可以通过该键盘和该指向装置将输入提供给计算机。其它种类的装置还可以用于提供与用户的交互;例如,提供给用户的反馈可以是任何形式的传感反馈(例如,视觉反馈、听觉反馈、或者触觉反馈);并且可以用任何形式(包括声输入、语音输入或者、触觉输入)来接收来自用户的输入。

[0128] 可以将此处描述的系统和技术实施在包括后台部件的计算系统(例如,作为数据服务器)、或者包括中间件部件的计算系统(例如,应用服务器)、或者包括前端部件的计算系统(例如,具有图形用户界面或者网络浏览器的用户计算机,用户可以通过该图形用户界面或者该网络浏览器来与此处描述的系统和技术实施方式交互)、或者包括这种后台部件、中间件部件、或者前端部件的任何组合的计算系统中。可以通过任何形式或者介质的数字数据通信(例如,通信网络)来将系统的部件相互连接。通信网络的示例包括:局域网(LAN)、广域网(WAN)和互联网。

[0129] 计算机系统可以包括客户端和服务端。客户端和服务端一般远离彼此并且通常通过通信网络进行交互。通过在相应的计算机上运行并且彼此具有客户端-服务器关系的计算机程序来产生客户端和服务端的关系。

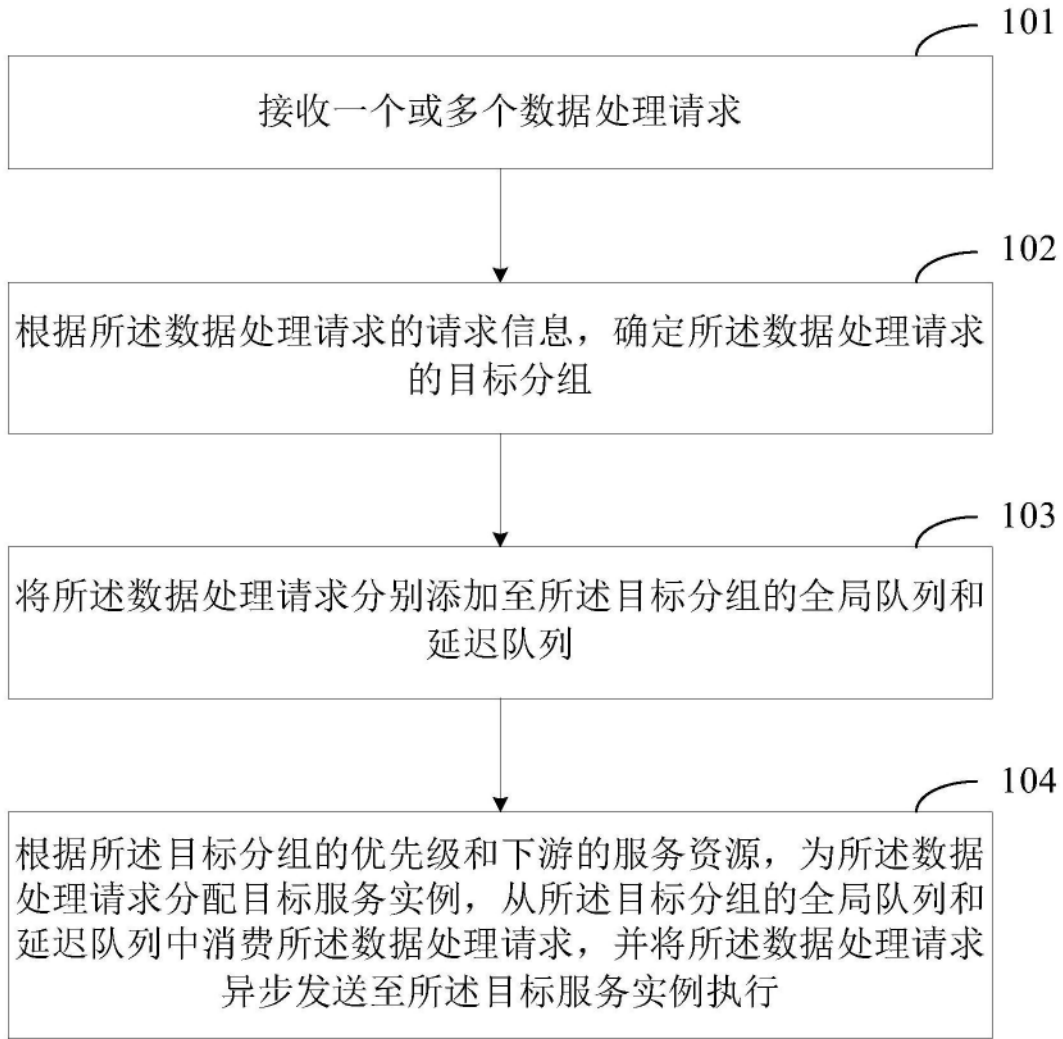


图1

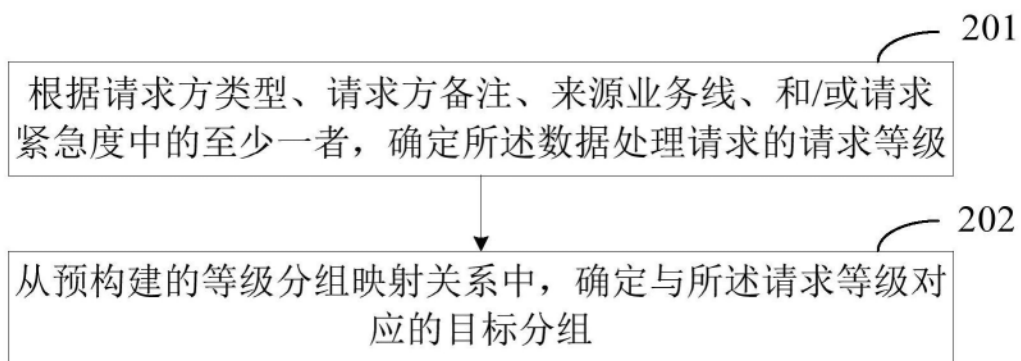


图2



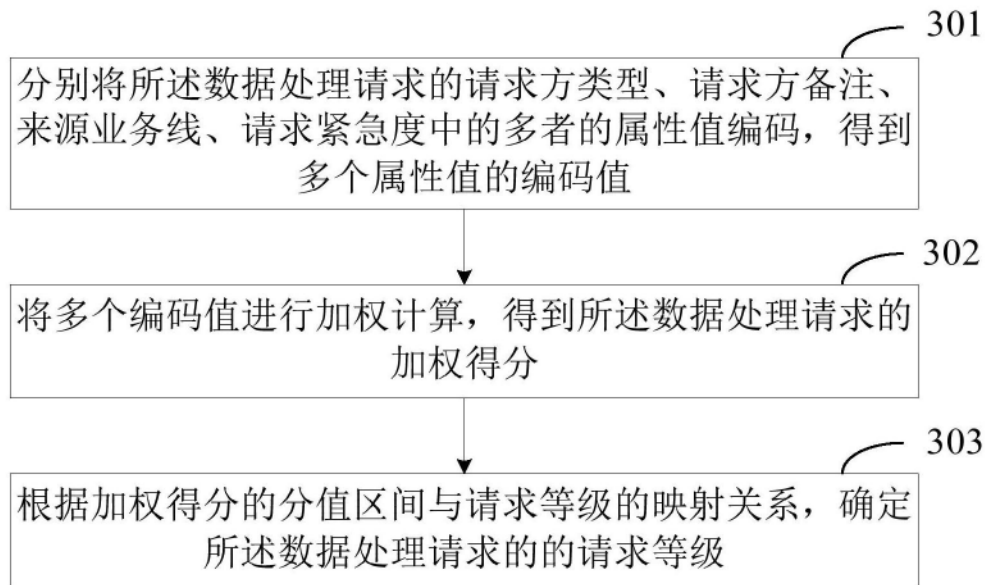


图3

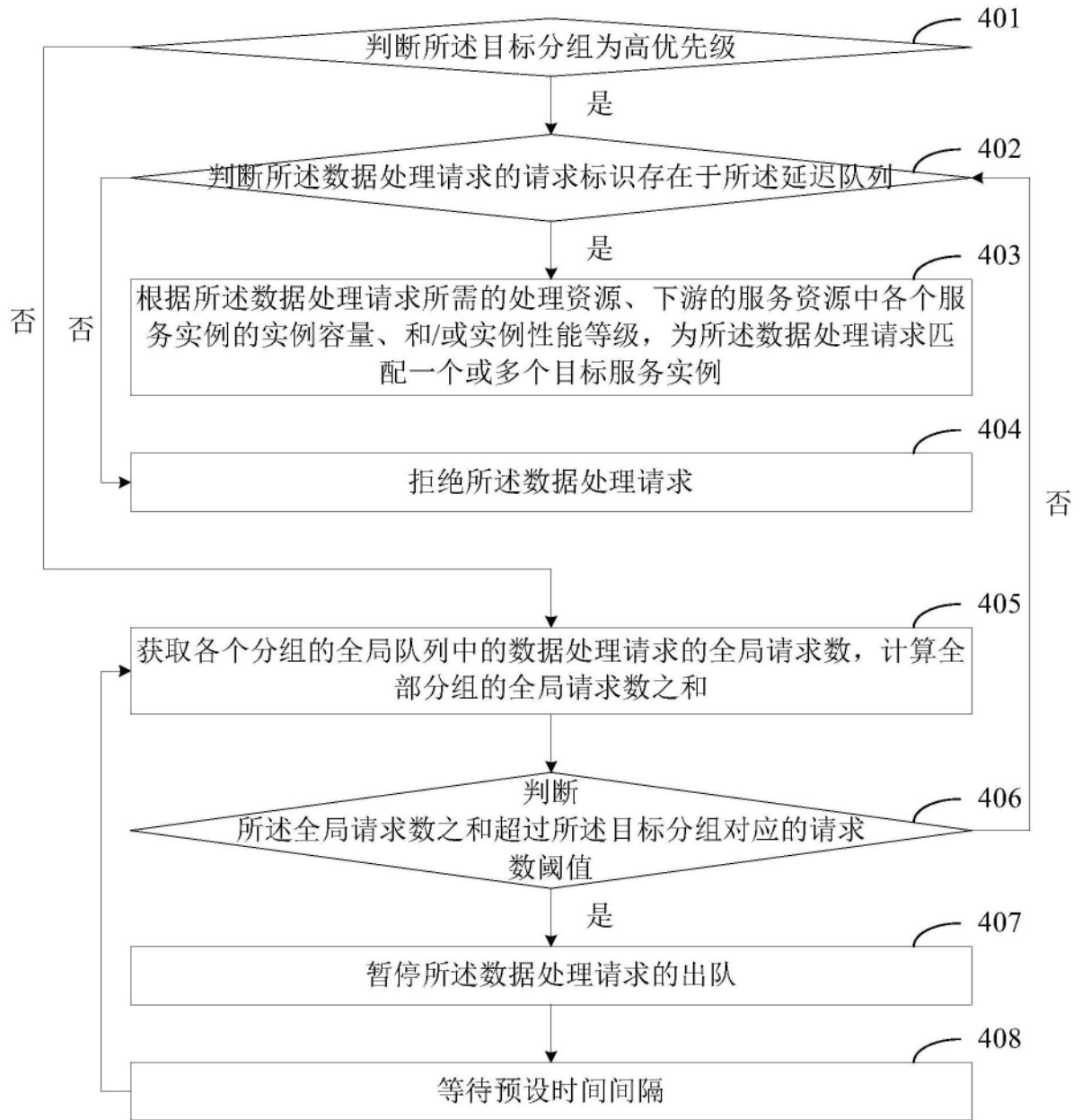


图4

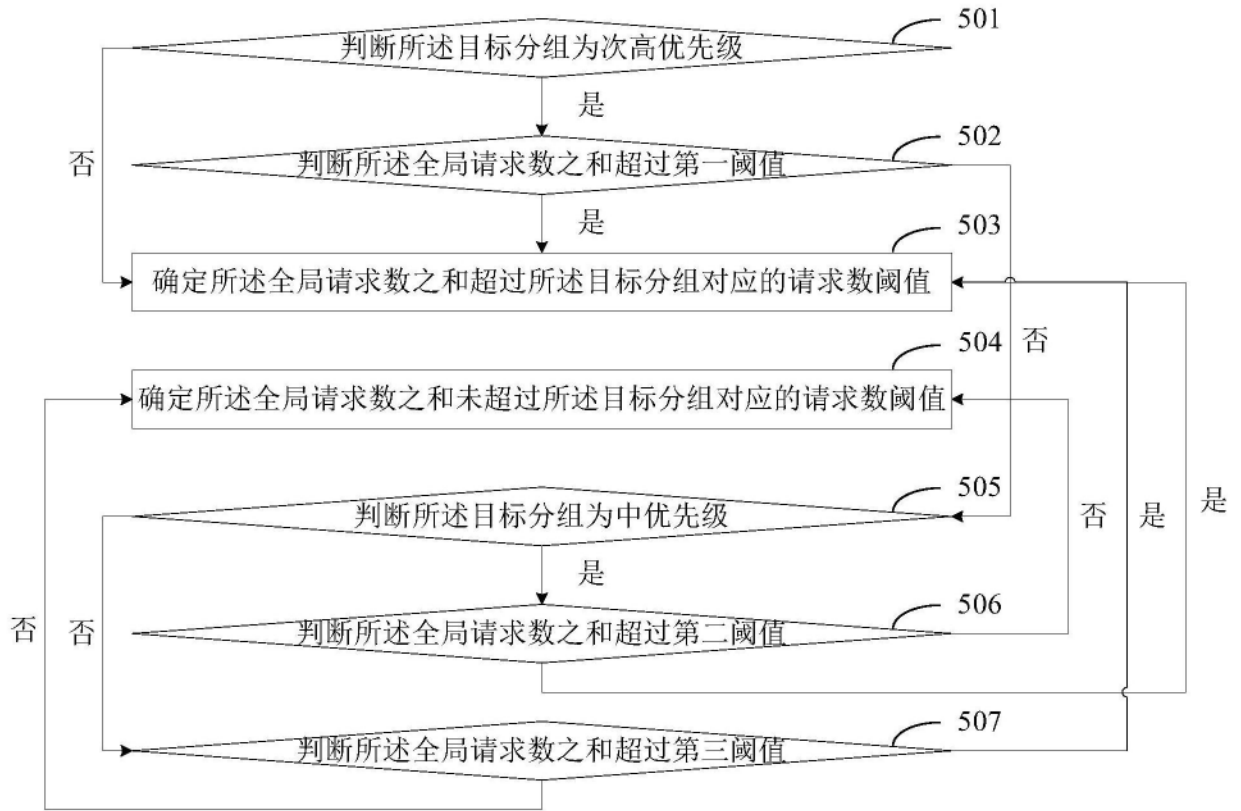


图5

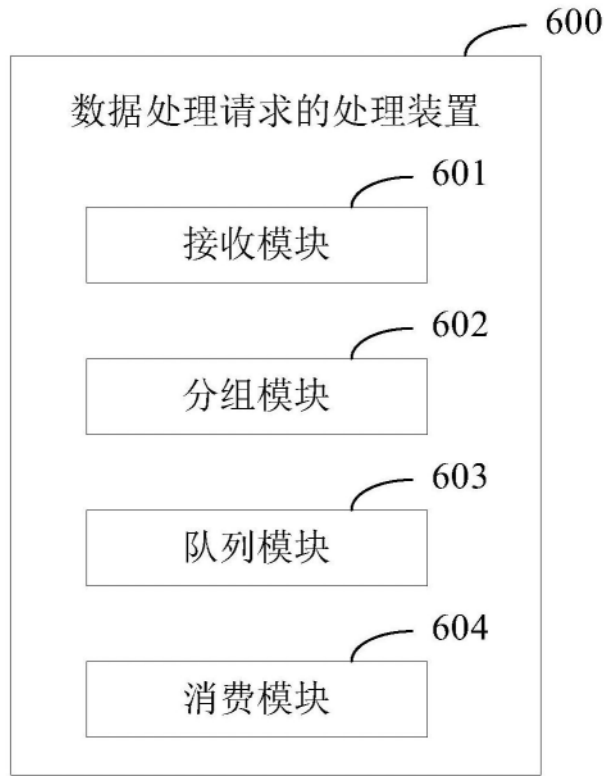


图6

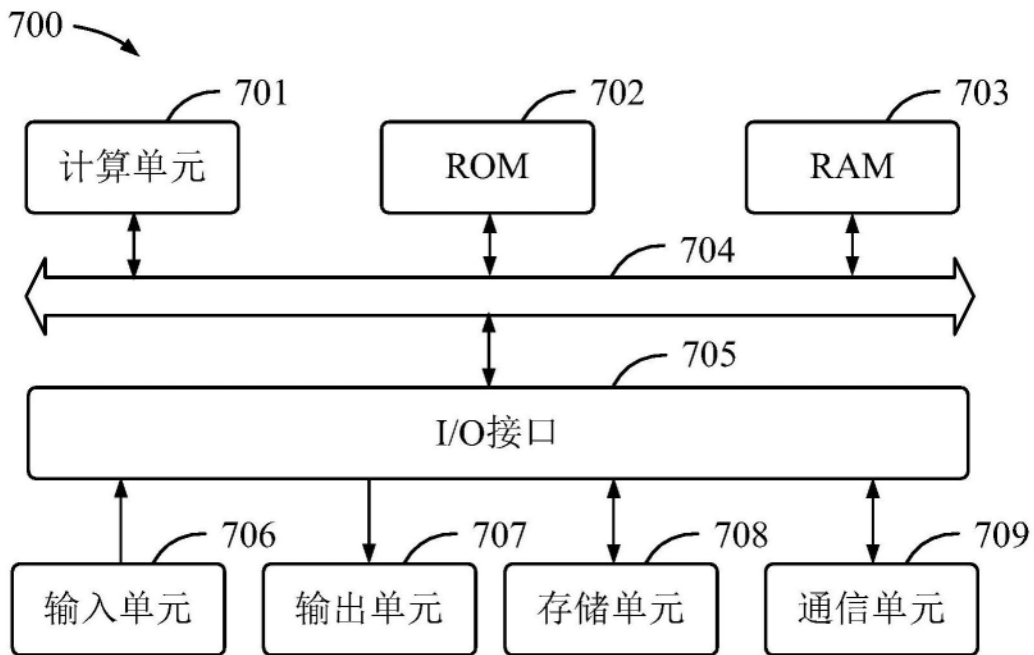


图7