(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104598563 B
(45)授权公告日 2018.09.04

(21)申请号 201510009667.2

(22)申请日 2015.01.08

(65)同一申请的已公布的文献号
申请公布号 CN 104598563 A

(43)申请公布日 2015.05.06

(73)专利权人 北京京东尚科信息技术有限公司
地址 100080 北京市海淀区杏石口路65号西杉创意园西区11C楼东段1-4层西段1-4层
专利权人 北京京东世纪贸易有限公司

(72)发明人 兰健

(74)专利代理机构 北京品源专利代理有限公司 11332
代理人 路凯 胡彬

(51)Int.Cl.
G06F 17/30(2006.01)
G06F 9/46(2006.01)

(56)对比文件
CN 1858735 A,2006.11.08,
CN 102622426 A,2012.08.01,
CN 103699660 A,2014.04.02,
US 2011/0302583 A1,2011.12.08,
CN 104102693 A,2014.10.15,

审查员 曾宪琴

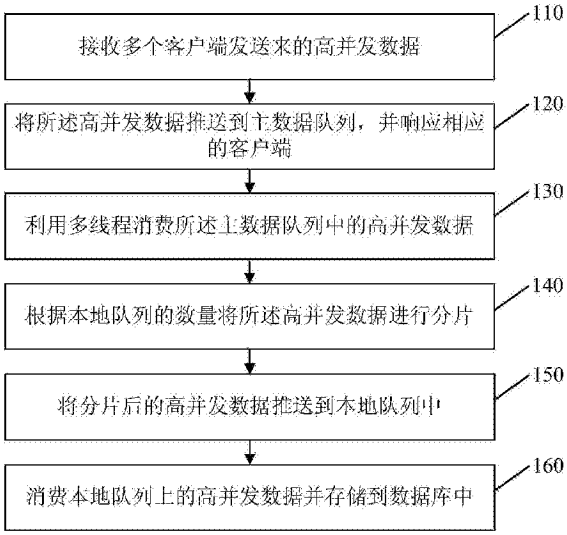
权利要求书1页 说明书5页 附图4页

(54)发明名称

高并发数据存储方法及装置

(57)摘要

本发明公开了一种高并发数据存储方法及装置。所述方法包括：接收多个客户端发送来的高并发数据；将所述高并发数据推送到主数据队列，并响应相应的客户端；利用多线程消费所述主数据队列中的高并发数据；根据本地队列的数量将所述高并发数据进行分片；将分片后的高并发数据推送到本地队列中；消费本地队列上的高并发数据并存储到数据库中。本发明通过主数据队列和本地队列对高并发数据进行缓存，采用了异步存储方式进行数据存储，缓解了数据库的压力，避免了数据库在高并发存储情况下宕机的问题，提升了高并发数据的存储效率。



CN 104598563 B

1.一种高并发数据存储方法，其特征在于，所述方法包括：

接收多个客户端发送来的高并发数据；

将所述高并发数据推送到主数据队列，并响应相应的客户端，其中，主数据队列是所有数据的主入口队列，当数据推送到该队列之后，立刻返回结果；

利用多线程消费所述主数据队列中的高并发数据；

将所述高并发数据的主键对本地队列的数量进行取模，其中，本地队列根据所述高并发数据的数据量进行横向扩展，当数据量大的时候扩展本地队列的数量；

根据相同的取模计算结果将对应的高并发数据分成一片；

根据取模计算结果将分片后的高并发数据推送到相应编号的本地队列中；

消费本地队列上的高并发数据并存储到数据库中。

2.根据权利要求1所述的方法，其特征在于，消费本地队列上的高并发数据并存储到数据库中，包括：

利用定时调度消费本地队列上的高并发数据；

将消费的高并发数据存储到数据库中。

3.根据权利要求2所述的方法，其特征在于，所述定时调度由Quartz实现。

4.一种高并发数据存储装置，其特征在于，所述装置包括：

接收模块，用于接收多个客户端发送来的高并发数据；

第一推送模块，用于将所述高并发数据推送到主数据队列，并响应相应的客户端，其中，主数据队列是所有数据的主入口队列，当数据推送到该队列之后，立刻返回结果；

消费模块，用于利用多线程消费所述主数据队列中的高并发数据；

分片模块，用于将所述高并发数据的主键对本地队列的数量进行取模；根据相同的取模计算结果将对应的高并发数据分成一片，其中，本地队列根据所述高并发数据的数据量进行横向扩展，当数据量大的时候扩展本地队列的数量；

第二推送模块，用于根据取模计算结果将分片后的高并发数据推送到相应编号的本地队列中；

存储模块，用于消费本地队列上的高并发数据并存储到数据库中。

5.根据权利要求4所述的装置，其特征在于，所述存储模块包括：

消费子模块，用于利用定时调度消费本地队列上的高并发数据；

存储子模块，用于将消费的高并发数据存储到数据库中。

6.根据权利要求5所述的装置，其特征在于，所述定时调度由Quartz实现。

高并发数据存储方法及装置

技术领域

[0001] 本发明实施例涉及数据处理技术，尤其涉及一种高并发数据存储方法及装置。

背景技术

[0002] 随着互联网信息的爆炸式增长，越来越多的互联网服务器将遭遇高并发、海量数据的环境，在这种环境下会面临高并发数据存储的问题。为了解决这个问题，Hadoop、MongoDB、Hbase等一些NoSql(Not Only SQL，非关系型数据库)语言得到了广泛的应用。

[0003] 但是，针对一些老业务系统，更换新的存储方式所消耗的成本会比较大，因此仍然使用着传统的关系型数据库存储数据，如MySql，Oracle等。当系统面临高并发数据存储的时候，会导致数据库的链接太多，占用资源过多，使得存储效率下降，进而导致服务器宕机等问题。

发明内容

[0004] 有鉴于此，本发明实施例提供一种高并发数据存储方法及装置，以提升高并发数据的存储效率。

[0005] 第一方面，本发明实施例提供了一种高并发数据存储方法，所述方法包括：

[0006] 接收多个客户端发送来的高并发数据；

[0007] 将所述高并发数据推送到主数据队列，并响应相应的客户端；

[0008] 利用多线程消费所述主数据队列中的高并发数据；

[0009] 根据本地队列的数量将所述高并发数据进行分片；

[0010] 将分片后的高并发数据推送到本地队列中；

[0011] 消费本地队列上的高并发数据并存储到数据库中。

[0012] 第二方面，本发明实施例还提供了一种高并发数据存储装置，所述装置包括：

[0013] 接收模块，用于接收多个客户端发送来的高并发数据；

[0014] 第一推送模块，用于将所述高并发数据推送到主数据队列，并响应相应的客户端；

[0015] 消费模块，用于利用多线程消费所述主数据队列中的高并发数据；

[0016] 分片模块，用于根据本地队列的数量将所述高并发数据进行分片；

[0017] 第二推送模块，用于将分片后的高并发数据推送到本地队列中；

[0018] 存储模块，用于消费本地队列上的高并发数据并存储到数据库中。

[0019] 本发明实施例通过将所述高并发数据推送到主数据队列，利用多线程消费所述主数据队列中的高并发数据，根据本地队列的数量将所述高并发数据进行分片，将分片后的高并发数据推送到本地队列中，消费本地队列上的高并发数据并存储到数据库中，通过主数据队列和本地队列对高并发数据进行缓存，采用了异步存储方式进行数据存储，缓解了直接将高并发数据存储到数据库时带给数据库的压力，避免了数据库在高并发存储情况下宕机的问题，提升了高并发数据的存储效率。

附图说明

[0020] 图1是本发明实施例一提供的高并发数据存储方法的流程图；

[0021] 图2是本发明实施例二提供的高并发数据存储方法的架构设计图；

[0022] 图3是本发明实施例二提供的高并发数据存储方法的流程图；

[0023] 图4是本发明实施例三提供的高并发数据存储装置的示意图。

具体实施方式

[0024] 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内容。

[0025] 实施例一

[0026] 图1是本发明实施例一提供的高并发数据存储方法的流程图，本实施例可适用于对高并发数据的存储，该方法可以由服务器来执行，具体包括如下步骤：

[0027] 步骤110，接收多个客户端发送来的高并发数据。

[0028] 其中，高并发是指在某一时刻访问量比较大。服务器接收多个客户端的请求，许多客户端同时请求形成高并发数据。

[0029] 步骤120，将所述高并发数据推送到主数据队列，并响应相应的客户端。

[0030] 其中，数据队列采用分布式MQ(Message Queue，消息队列)，可以支持分布式扩展，也可以使得框架具有高可用性，在大数据处理的时候仍可具有较为客观的性能。其中，ActiveMQ是Apache出品的，最流行的、能力强劲的开源消息总线。

[0031] 服务器接收到高并发数据后，将所述高并发数据推送到主数据队列，并响应发送该高并发数据的客户端。其中，主数据队列是所有数据的主入口队列，当数据推送到该队列之后，立刻返回结果，提升响应速度。

[0032] 步骤130，利用多线程消费所述主数据队列中的高并发数据。

[0033] 其中，多线程由线程池进行分配，线程池提供多线程处理数据并将数据进行分片推送到对应的本地队列，提升消息的消费速度，同时还能利用MQ的特性，持久化数据，防止数据的丢失。服务器利用线程池分配的多线程快速消费所述主数据队列中的高并发数据。其中，消费所述主数据队列中的高并发数据是指将主数据队列中的高并发数据移出该主数据队列。

[0034] 步骤140，根据本地队列的数量将所述高并发数据进行分片。

[0035] 其中，本地队列用于存放分片后的高并发数据，防止分片后的数据丢失。本地队列可以根据所述高并发数据的数据量进行横向扩展，当数据量大的时候可以扩展本地队列的数量。服务器根据本地队列的数量，将从主数据队列中移出的高并发数据分成与本地队列相同数量的片段，以将分片后的高并发数据存储到本地队列中。对高并发数据进行分片处理，保证数据处理不重复，且提升处理速度。本领域技术人员可以了解，对高并发数据进行分片有多种策略：可以根据数据的主键特征进行分片，将高并发数据的主键对本地队列的数量进行取模计算，得到的结果是几就将该数据分配到编号为几的本地队列中；也可以根据时间范围进行分片，例如：可以将前100万个数据分配到第一个本地队列中，将第二个100

万个数据分配到第二个本地队列中；当然，还有其他分片策略。

[0036] 步骤150，将分片后的高并发数据推送到本地队列中。

[0037] 服务器根据对高并发数据进行分片的分片策略，将分片后的高并发数据推送到相应的本地队列中，由本地队列对高并发数据进行缓存。

[0038] 步骤160，消费本地队列上的高并发数据并存储到数据库中。

[0039] 服务器消费本地队列上的高并发数据，即将本地队列上的高并发数据移出本地队列，然后将移出本地队列的高并发数据存储到数据库中。

[0040] 本实施例中的高并发数据在传递过程(例如：将高并发数据推送到主数据队列、消费数据等)中，可以采用高性能的JSON格式进行传递，方便数据的序列化和反序列化，使得数据的传递速度更快。其中，JSON(JavaScript Object Notation)是一种轻量级的数据交换格式，是基于JavaScript语法的子集，即数组和对象表示。序列化是一种用来处理对象流的机制，对象流是将对象的内容进行流化，可以对流化后的对象进行读写操作，序列化是为了解决对象流进行读写操作时引发的问题的。

[0041] 本实施例通过将所述高并发数据推送到主数据队列，利用多线程消费所述主数据队列中的高并发数据，根据本地队列的数量将所述高并发数据进行分片，将分片后的高并发数据推送到本地队列中，消费本地队列上的高并发数据并存储到数据库中，通过主数据队列和本地队列对高并发数据进行缓存，采用了异步存储方式进行数据存储，缓解了直接将高并发数据存储到数据库时带给数据库的压力，避免了数据库在高并发存储情况下宕机的问题，提升了高并发数据的存储效率。

[0042] 在上述技术方案的基础上，根据本地队列的数量将所述高并发数据进行分片，优选包括：

[0043] 将所述高并发数据的主键对所述本地队列的数量进行取模；

[0044] 根据相同的取模计算结果将对应的高并发数据分成一片。

[0045] 首先将高并发数据中生成的主键对所述本地队列的数量进行取模计算，将得到相同取模计算结果的高并发数据分成一个片段。

[0046] 在上述技术方案的基础上，将分片后的高并发数据推送到本地队列中，具体包括：

[0047] 根据取模计算结果将分片后的高并发数据推送到相应编号的本地队列中。

[0048] 其中，每个本地队列有自己的编号。将分片后的高并发数据推送到与取模计算结果相同编号的本地队列中。

[0049] 在上述任一技术方案的基础上，消费本地队列上的高并发数据并存储到数据库中，优选包括：

[0050] 利用定时调度消费本地队列上的高并发数据；

[0051] 将消费的高并发数据存储到数据库中。

[0052] 服务器利用定时调度将本地队列上的高并发数据定时/即时移出本地队列，用户可以根据需求控制抓取数据(移出本地队列)的数量和抓取数据的时间间隔，将定时/即时移出本地队列的高并发数据存储到数据库中。通过定时调度可以配置定时/即时处理数据的功能，缓解数据库的存储压力。优选的，所述定时调度由Quartz实现。其中，Quartz是一个完全由Java编写的开源作业调度框架。

[0053] 实施例二

[0054] 图2是本发明实施例二提供的高并发数据存储方法的架构设计图。如图2所示，主数据队列是所有数据的主入口队列，当数据推送到该队列之后，立刻返回结果，提升响应速度；线程池(Thread Pool)提供多线程处理数据并将数据进行分片推送到对应的本地队列，提升消息的消费速度，同时还能利用MQ的特性，持久化数据，防止数据的丢失；本地队列用于存放数据分片后的数据，防止分片后的数据丢失；工作(Work)1、工作2、工作3通过定时/即时消费数据，处理数据并存储到数据库中，可以自主控制抓取数据的数量和抓取数据的时间间隔。

[0055] 图3是本发明实施例二提供的高并发数据存储方法的流程图。如图3所示，本实施例提供的高并发数据存储方法具体包括如下步骤：

[0056] 步骤310，将待存储的高并发数据推送到主数据队列。

[0057] 首先将待存储的高并发数据推送到主数据队列，当高并发数据推送到主数据队列后，返回结果，响应客户端。

[0058] 步骤320，多线程消费主数据队列中的高并发数据。

[0059] 利用线程池提供的多线程快速消费主数据队列中的高并发数据。

[0060] 步骤330，将高并发数据进行分片并推送到对应的本地队列中。

[0061] 根据本地队列的数量，将高并发数据的主键对本地队列的数量进行取模计算，按照取模计算结果将分片后的高并发数据推送到编号为取模计算结果的本地队列中。

[0062] 步骤340，定时消费本地队列中的高并发数据。

[0063] 利用Quartz实现定时/即时消费本地队列中的高并发数据。

[0064] 步骤350，存储数据到数据库。

[0065] 本实施例针对传统的关系型数据库，利用主数据队列对高并发数据进行缓存，利用本地队列对高并发数据进行二次数据缓存，可以防止数据丢失，也能够提升主数据队列的消费速度，提升了高并发数据的存储效率，可以配置定时/即时处理数据的功能，缓解了数据库的压力，避免了数据库在高并发存储情况下宕机的问题。

[0066] 实施例三

[0067] 图4是本发明实施例三提供的高并发数据存储装置的示意图。本实施例提供的高并发数据存储装置用于实现实施例一提供的高并发数据存储方法。如图4所示，本实施例提供的高并发数据存储装置包括：接收模块410、第一推送模块420、消费模块430、分片模块440、第二推送模块450和存储模块460。

[0068] 其中，接收模块410用于接收多个客户端发送来的高并发数据；第一推送模块420用于将所述高并发数据推送到主数据队列，并响应相应的客户端；消费模块430用于利用多线程消费所述主数据队列中的高并发数据；分片模块440用于根据本地队列的数量将所述高并发数据进行分片；第二推送模块450用于将分片后的高并发数据推送到本地队列中；存储模块460用于消费本地队列上的高并发数据并存储到数据库中。

[0069] 优选的，所述分片模块包括：取模子模块，用于将所述高并发数据的主键对所述本地队列的数量进行取模；分片子模块，用于根据相同的取模计算结果将对应的高并发数据分成一片。优选的，所述第二推送模块具体用于根据取模计算结果将分片后的高并发数据推送到相应编号的本地队列中。

[0070] 优选的，所述存储模块包括：消费子模块，用于利用定时调度消费本地队列上的高

并发数据；存储子模块，用于将消费的高并发数据存储到数据库中。有选的，所述定时调度由Quartz实现。

[0071] 本实施例通过接收模块接收多个客户端发送来的高并发数据；第一推送模块将所述高并发数据推送到主数据队列，并响应相应的客户端；消费模块利用多线程消费所述主数据队列中的高并发数据；分片模块根据本地队列的数量将所述高并发数据进行分片；第二推送模块将分片后的高并发数据推送到本地队列中；存储模块消费本地队列上的高并发数据并存储到数据库中。通过主数据队列和本地队列对高并发数据进行缓存，采用了异步存储方式进行数据存储，缓解了直接将高并发数据存储到数据库时带给数据库的压力，避免了数据库在高并发存储情况下宕机的问题，提升了高并发数据的存储效率。

[0072] 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。

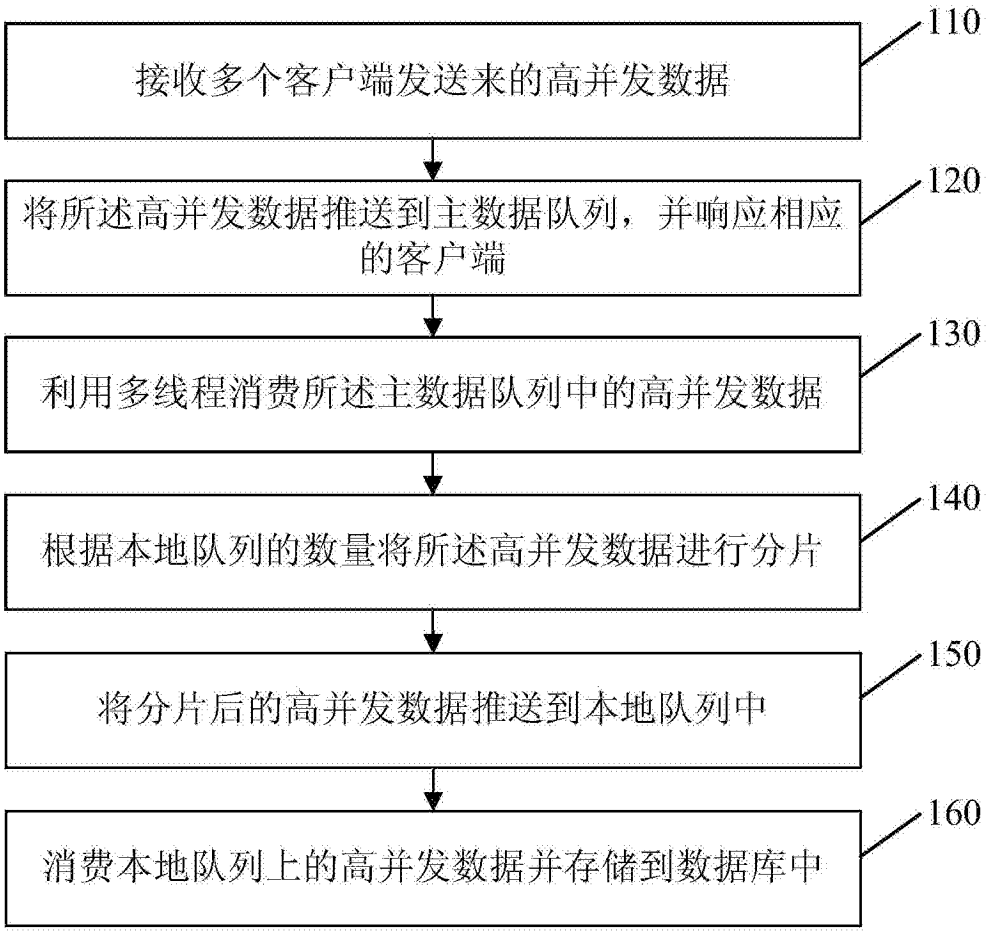

接收多个客户端发送来的高并发数据
110
将所述高并发数据推送到主数据队列，并响应相应的客户端
120
利用多线程消费所述主数据队列中的高并发数据
130
根据本地队列的数量将所述高并发数据进行分片
140
将分片后的高并发数据推送到本地队列中
150
消费本地队列上的高并发数据并存储到数据库中
160

图1

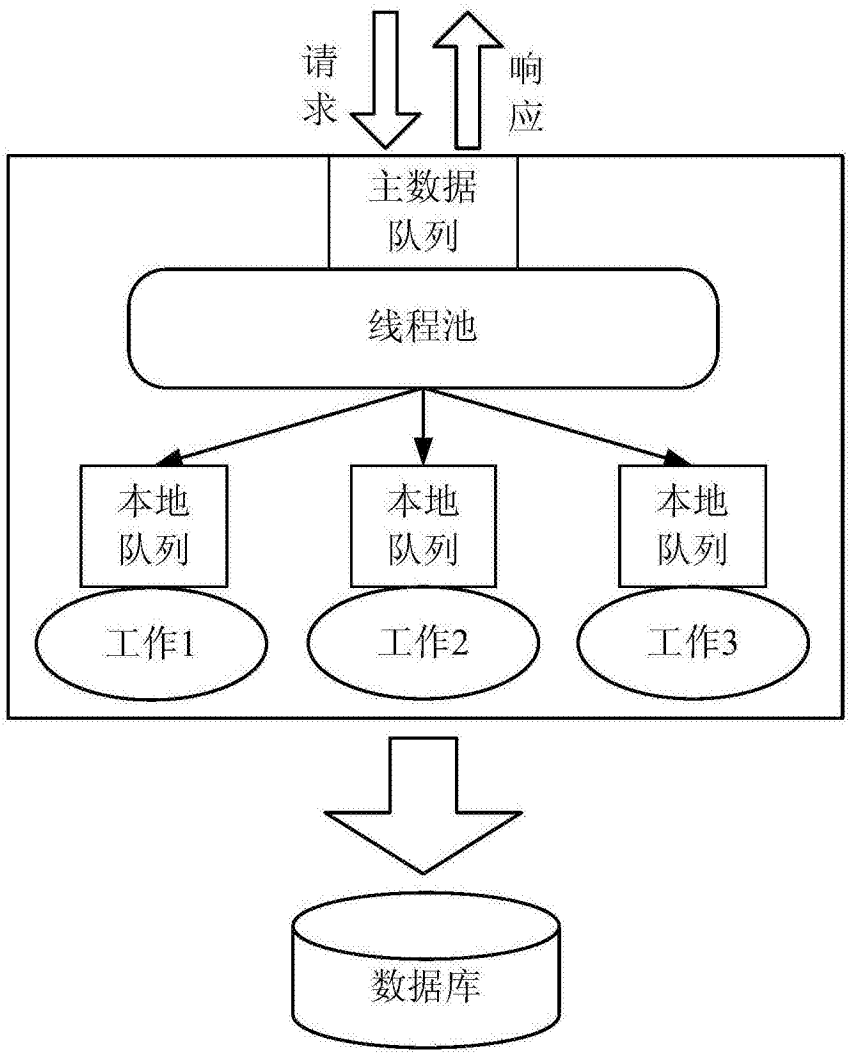


图2

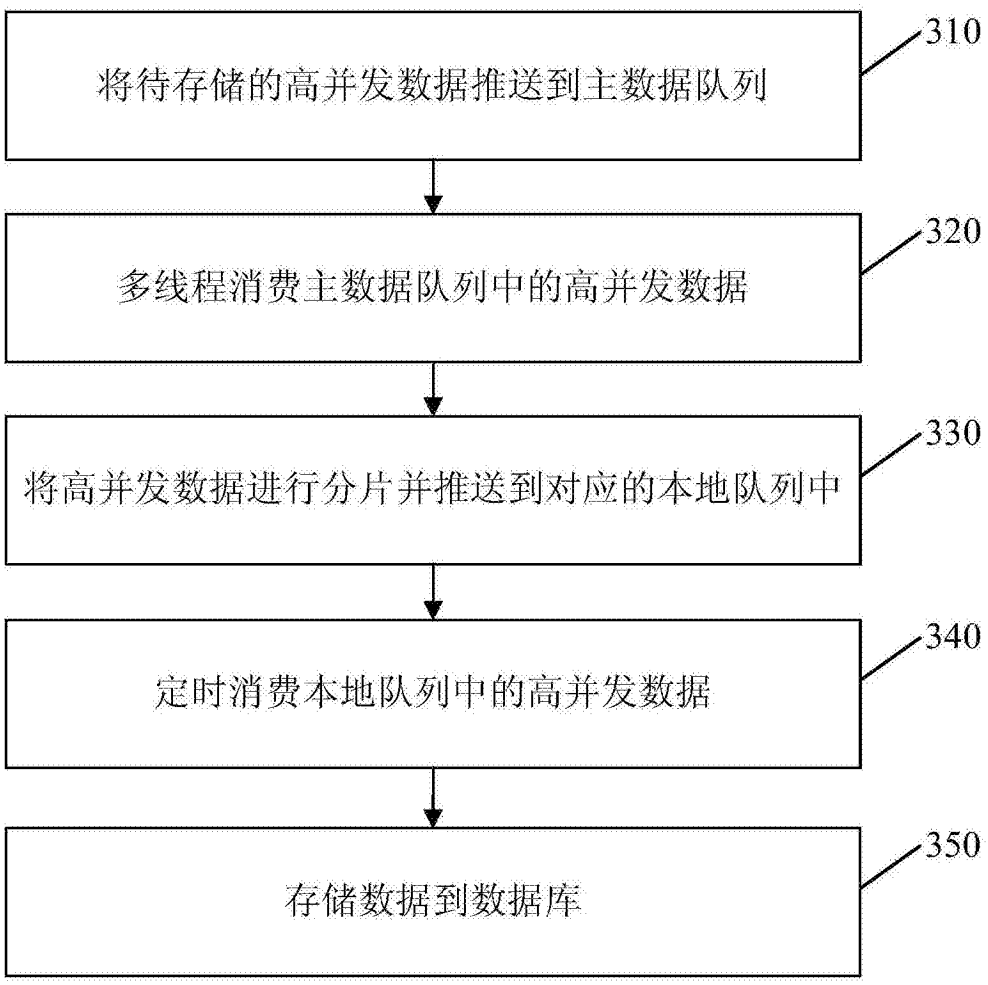

将待存储的高并发数据推送到主数据队列
310
多线程消费主数据队列中的高并发数据
320
将高并发数据进行分片并推送到对应的本地队列中
330
定时消费本地队列中的高并发数据
340
存储数据到数据库
350

图3

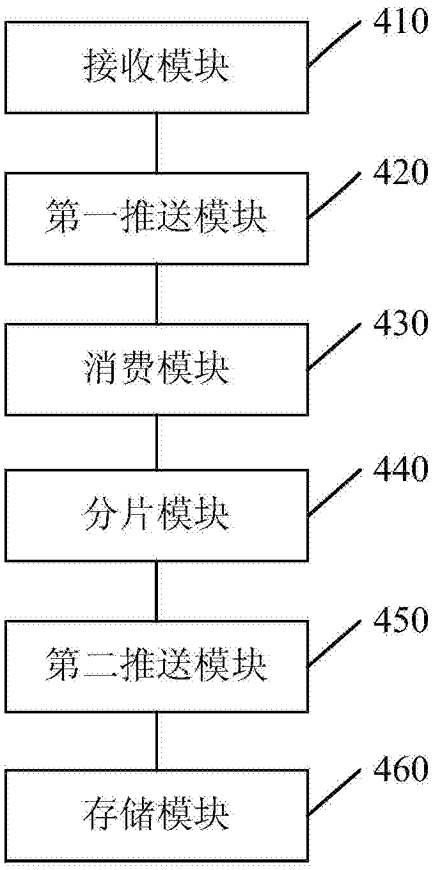


图4