(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111190537 B
(45)授权公告日 2023.08.25

(21)申请号 201911278697.8

(22)申请日 2019.12.10

(65)同一申请的已公布的文献号
申请公布号 CN 111190537 A

(43)申请公布日 2020.05.22

(73)专利权人 优刻得科技股份有限公司
地址 200093 上海市杨浦区隆昌路619号10#楼B座

(72)发明人 蔡军 方然

(74)专利代理机构 北京集佳知识产权代理有限公司 11227
专利代理师 古利兰

(51)Int.Cl.
G06F 3/06(2006.01)

(56)对比文件
CN 102169419 A,2011.08.31
CN 102779180 A,2012.11.14
CN 104035729 A,2014.09.10
CN 105630808 A,2016.06.01
CN 108958660 A,2018.12.07
CN 109614054 A,2019.04.12
JP 2006331076 A,2006.12.07
JP 2008140396 A,2008.06.19
JP 2009116946 A,2009.05.28
US 2014289493 A1,2014.09.25

审查员 马银银

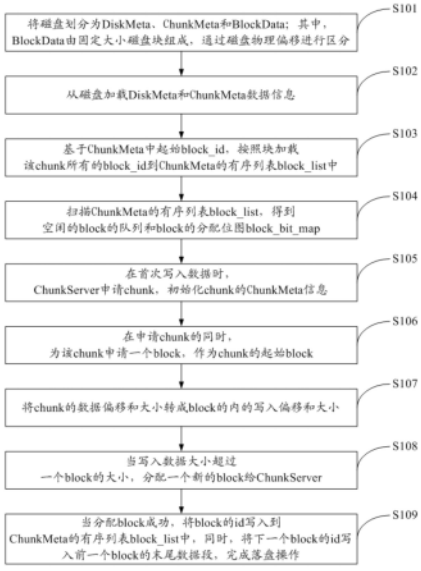
权利要求书3页 说明书9页 附图6页

(54)发明名称

一种追加写场景下顺序存储磁盘管理的方法及系统

(57)摘要

本发明公开了一种追加写场景下顺序存储磁盘管理的方法及系统，方法包括：首先定义磁盘结构，对磁盘初始化，在首次写入数据时，ChunkServer申请chunk，初始化chunk的ChunkMeta信息；在申请chunk的同时，为该chunk申请一个block，作为chunk的起始block；将chunk的数据偏移和大小转成block的内的写入偏移和大小；当写入数据大小超过一个block的大小，分配一个新的block给ChunkServer；当分配block成功，将block的id写入到ChunkMeta的有序列表block_list中，同时，将下一个block的id写入前一个block的末尾数据段，完成落盘操作。本发明能够有效的对裸盘按照块的方式进行管理，灵活定义块的大小，利用追加写的办法提升存储速度。



CN 111190537 B

1.一种追加写场景下顺序存储磁盘管理的方法,其特征在于,包括:

将磁盘划分为层磁盘元数据段DiskMeta、chunk原数据段ChunkMeta和BlockData;其中,所述BlockData由固定大小磁盘块组成,通过磁盘物理偏移进行区分;所述BlockData为自定义名称,表示Block的数据块叫做BlockData;所述DiskMeta为存放磁盘的元数据信息,至少包括磁盘的大小、磁盘中的block个数;所述ChunkMeta为存储chunk的元信息,包括chunk的index,chunk已经使用的长度、可使用的长度,已经分配的磁盘的block的id;所述chunk为自定义名称,表示分布式存储系统对外开放提供的最小存储单位;所述block为自定义名称,表示分布式存储系统的单机上会将磁盘均匀划分成N个块,每一个块称之为一个block;

从磁盘加载DiskMeta和ChunkMeta数据信息;

基于所述ChunkMeta中起始block_id,按照块加载该chunk所有的block_id到所述ChunkMeta的有序列表block_list中;其中,所述block_id表示已经分配的磁盘的block的id;

扫描所述ChunkMeta的有序列表block_list,得到空闲的block的队列和block的分配位图block_bit_map;

在首次写入数据时,ChunkServer申请chunk,初始化chunk的ChunkMeta信息;其中,所述ChunkServer为自定义名称,是一种服务,管理chunk、ChunkMeta、Block的服务,负责接收和处理Client的文件存储请求;

在申请chunk的同时,为该chunk申请一个block,作为chunk的起始block;

将chunk的数据偏移和大小转成block的内的写入偏移和大小;

当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer;

当分配block成功,将block的id写入到所述ChunkMeta的有序列表block_list中,同时,将下一个block的id写入前一个block的末尾数据段,完成落盘操作。

2.根据权利要求1所述的方法,其特征在于,所述当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer,包括:

判断空闲block队列free_block_list是否为空,若是,则返回磁盘满的状态码,若否,则:

从所述空闲block队列free_block_list中获取第一个block;

通过所述block的分配位图block_bit_map判断该block是否被使用,若否,则:

该block分配成功,将所述block的分配位图block_bit_map的分配的位置置1。

3.根据权利要求2所述的方法,其特征在于,还包括:

按照chunk上的偏移计算读取数据在chunk的第几个block上;

基于计算结果,在所述ChunkMeta中的有序列表block_list中,获取block的id;

按照block的id计算磁盘上的偏移地址,按照偏移读取数据;

判断读取数据大小是否超过当前的block,若否,则读取结束返回数据,若是,则从所述ChunkMeta中的有序列表block_list中获取下一个block。

4.根据权利要求3所述的方法,其特征在于,还包括:

删除ChunkMeta元数据;

当删除所述ChunkMeta元数据成功时,释放block块。

5.根据权利要求4所述的方法,其特征在于,当删除所述ChunkMeta元数据成功时,释放block块,包括:

将block的id加入空闲block队列free_block_list末尾;

将所述block的分配位图block_bit_map的分配位置置0。

6.一种追加写场景下顺序存储磁盘管理的系统,其特征在于,包括:

定义模块,用于将磁盘划分为层磁盘元数据段DiskMeta、chunk原数据段ChunkMeta和BlockData;其中,所述BlockData由固定大小磁盘块组成,通过磁盘物理偏移进行区分;所述BlockData为自定义名称,表示Block的数据块叫做BlockData;所述DiskMeta为存放磁盘的元数据信息,至少包括磁盘的大小、磁盘中的block个数;所述ChunkMeta为存储chunk的元信息,包括chunk的index,chunk已经使用的长度、可使用的长度,已经分配的磁盘的block的id;所述chunk为自定义名称,表示分布式存储系统对外开放提供的最小存储单位;所述block为自定义名称,表示分布式存储系统的单机上会将磁盘均匀划分成N个块,每一个块称之为一个block;

第一加载模块,用于从磁盘加载DiskMeta和ChunkMeta数据信息;

第二加载模块,用于基于所述ChunkMeta中起始block_id,按照块加载该chunk所有的block_id到所述ChunkMeta的有序列表block_list中;其中,所述block_id表示已经分配的磁盘的block的id;

扫描模块,用于扫描所述ChunkMeta的有序列表block_list,得到空闲的block的队列和block的分配位图block_bit_map;

初始化模块,用于在首次写入数据时,ChunkServer申请chunk,初始化chunk的ChunkMeta信息;其中,所述ChunkServer为自定义名称,是一种服务,管理chunk、ChunkMeta、Block的服务,负责接收和处理Client的文件存储请求;

申请模块,用于在申请chunk的同时,为该chunk申请一个block,作为chunk的起始block;

转换模块,用于将chunk的数据偏移和大小转成block的内的写入偏移和大小;

分配模块,用于当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer;

写入模块,用于当分配block成功,将block的id写入到所述ChunkMeta的有序列表block_list中,同时,将下一个block的id写入前一个block的末尾数据段,完成落盘操作。

7.根据权利要求6所述的系统,其特征在于,所述分配模块在执行当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer时,包括:

第一判断单元,用于判断空闲block队列free_block_list是否为空;

返回单元,用于当空闲block队列free_block_list为空时,返回磁盘满的状态码;

获取单元,用于当空闲block队列free_block_list不为空时,从所述空闲block队列free_block_list中获取第一个block;

第二判断单元,用于通过所述block的分配位图block_bit_map判断该block是否被使用;

设置单元,用于当通过所述block的分配位图block_bit_map判断该block未被使用时,该block分配成功,将所述block的分配位图block_bit_map的分配的位置置1。

8.根据权利要求7所述的系统,其特征在于,还包括:

计算模块,用于按照chunk上的偏移计算读取数据在chunk的第几个block上;

第一获取模块,用于基于计算结果,在所述ChunkMeta中的有序列表block_list中,获取block的id;

读取模块,用于按照block的id计算磁盘上的偏移地址,按照偏移读取数据;

判断模块,用于判断读取数据大小是否超过当前的block;

返回模块,用于当读取数据大小未超过是否当前的block时,读取结束返回数据;

第二获取模块,用于当读取数据大小未超过当前的block时,从所述ChunkMeta中的有序列表block_list中获取下一个block。

9.根据权利要求8所述的系统,其特征在于,还包括:

删除模块,用于删除ChunkMeta元数据;

释放模块,用于当删除所述ChunkMeta元数据成功时,释放block块。

10.根据权利要求9所述的系统,其特征在于,所述释放模块具体用于:

将block的id加入空闲block队列free_block_list末尾;

将所述block的分配位图block_bit_map的分配位置置0。

一种追加写场景下顺序存储磁盘管理的方法及系统

技术领域

[0001] 本发明涉及磁盘管理技术领域,尤其涉及一种追加写场景下顺序存储磁盘管理的方法及系统。

背景技术

[0002] 随着SSD(Solid State Disk,固态硬盘)等存储介质的出现,磁盘IO效率得到巨大突破,存储方式已经成了云存储瓶颈优化的重点。顺序读写是提升存储速度的一种重要方式,追加写是实现顺序读写的一种方式。在现有技术中,对于单机磁盘数据管理通常由两种方式,一种是使用本地文件系统进行数据管理;另外一种是对数据块进行数据管理,在系统初始化时,将磁盘划分成固定的大小块(通常被称为chunk),系统在存储文件数据时,通过算法分配磁盘上多个chunk(分布式存储系统对外开放提供的最小存储单位)来存储该文件,记录chunk与文件的对应关系,最后,将对应的关系持久化到数据盘或者数据库中。

[0003] 现有分布式存储场景下的单机磁盘处理技术的主要缺陷有以下几点:

[0004] 1、若chunk设置过大,系统事先会为每个chunk分配出固定大小,当存储小文件的时候,就会造成很大的浪费。例如,对于一个1G的chunk,实际未写满1GB,仍然会占用1GB的空间。

[0005] 2、若chunk设置过小,系统为了快速索引chunk,会缓存chunk的上下文信息,导致chunk上下文信息占据较多的内存空间,而且每一次分配一个chunk需要持久化一次元数据,持久化操作的开销较大,不适合高性能的存储系统。

[0006] 3、若使用本地文件系统存储,在IO路径上需要对文件元数据进行读写,导致写放大,耗费大量的CPU和磁盘IO资源,并且本地文件系统在某些场景下存在断电丢数据的风险。

[0007] 4、数据文件存储的位置靠文件维持,元数据丢失的情况下,数据存储的位置无法恢复。

[0008] 因此,如何有效的对追加写场景下顺序存储磁盘进行管理,是一项亟待解决的问题。

发明内容

[0009] 有鉴于此,本发明提供了一种追加写场景下顺序存储磁盘管理的方法,能够有效的对裸盘按照块的方式进行管理,灵活定义块的大小,利用追加写的办法提升存储速度。

[0010] 本发明提供了一种追加写场景下顺序存储磁盘管理的方法,包括:

[0011] 将磁盘划分为DiskMeta、ChunkMeta和BlockData;其中,所述BlockData由固定大小磁盘块组成,通过磁盘物理偏移进行区分;

[0012] 从磁盘加载DiskMeta和ChunkMeta数据信息;

[0013] 基于所述ChunkMeta中起始block_id,按照块加载该chunk所有的block_id到所述ChunkMeta的有序列表block_list中;

[0014] 扫描所述ChunkMeta的有序列表block_list,得到空闲的block的队列和block的分配位图block_bit_map;

[0015] 在首次写入数据时,ChunkServer申请chunk,初始化chunk的ChunkMeta信息;

[0016] 在申请chunk的同时,为该chunk申请一个block,作为chunk的起始block;

[0017] 将chunk的数据偏移和大小转成block的内的写入偏移和大小;

[0018] 当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer;

[0019] 当分配block成功,将block的id写入到所述ChunkMeta的有序列表block_list中,同时,将下一个block的id写入前一个block的末尾数据段,完成落盘操作。

[0020] 优选地,所述当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer,包括:

[0021] 判断空闲block队列free_block_list是否为空,若是,则返回磁盘满的状态码,若否,则:

[0022] 从所述空闲block队列free_block_list中获取第一个block;

[0023] 通过所述block的分配位图block_bit_map判断该block是否被使用,若否,则:

[0024] 该block分配成功,将所述block的分配位图block_bit_map的分配的位置置1。

[0025] 优选地,所述方法还包括:

[0026] 按照chunk上的偏移计算读取数据在chunk的第几个block上;

[0027] 基于计算结果,在所述ChunkMeta中的有序列表block_list中,获取block的id;

[0028] 按照block的id计算磁盘上的偏移地址,按照偏移读取数据;

[0029] 判断读取数据大小是否超过当前的block,若否,则读取结束返回数据,若是,则从所述ChunkMeta中的有序列表block_list中获取下一个block。

[0030] 优选地,所述方法还包括:

[0031] 删除ChunkMeta元数据;

[0032] 当删除所述ChunkMeta元数据成功时,释放block块。

[0033] 优选地,当删除所述ChunkMeta元数据成功时,释放block块,包括:

[0034] 将block的id加入空闲block队列free_block_list末尾;

[0035] 将所述block的分配位图block_bit_map的分配位置置0。

[0036] 一种追加写场景下顺序存储磁盘管理的系统,包括:

[0037] 定义模块,用于将磁盘划分为DiskMeta、ChunkMeta和BlockData;其中,所述BlockData由固定大小磁盘块组成,通过磁盘物理偏移进行区分;

[0038] 第一加载模块,用于从磁盘加载DiskMeta和ChunkMeta数据信息;

[0039] 第二加载模块,用于基于所述ChunkMeta中起始block_id,按照块加载该chunk所有的block_id到所述ChunkMeta的有序列表block_list中;

[0040] 扫描模块,用于扫描所述ChunkMeta的有序列表block_list,得到空闲的block的队列和block的分配位图block_bit_map;

[0041] 初始化模块,用于在首次写入数据时,ChunkServer申请chunk,初始化chunk的ChunkMeta信息;

[0042] 申请模块,用于在申请chunk的同时,为该chunk申请一个block,作为chunk的起始block;

[0043] 转换模块,用于将chunk的数据偏移和大小转成block的内的写入偏移和大小;

[0044] 分配模块,用于当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer;

[0045] 写入模块,用于当分配block成功,将block的id写入到所述ChunkMeta的有序列表block_list中,同时,将下一个block的id写入前一个block的末尾数据段,完成落盘操作。

[0046] 优选地,所述分配模块在执行当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer时,包括:

[0047] 第一判断单元,用于判断空闲block队列free_block_list是否为空;

[0048] 返回单元,用于当空闲block队列free_block_list为空时,返回磁盘满的状态码;

[0049] 获取单元,用于当空闲block队列free_block_list不为空时,从所述空闲block队列free_block_list中获取第一个block;

[0050] 第二判断单元,用于通过所述block的分配位图block_bit_map判断该block是否被使用;

[0051] 设置单元,用于当通过所述block的分配位图block_bit_map判断该block未被使用时,该block分配成功,将所述block的分配位图block_bit_map的分配的位置置1。

[0052] 优选地,所述系统还包括:

[0053] 计算模块,用于按照chunk上的偏移计算读取数据在chunk的第几个block上;

[0054] 第一获取模块,用于基于计算结果,在所述ChunkMeta中的有序列表block_list中,获取block的id;

[0055] 读取模块,用于按照block的id计算磁盘上的偏移地址,按照偏移读取数据;

[0056] 判断模块,用于判断读取数据大小是否超过当前的block;

[0057] 返回模块,用于当读取数据大小未超过是否当前的block时,读取结束返回数据;

[0058] 第二获取模块,用于当读取数据大小未超过当前的block时,从所述ChunkMeta中的有序列表block_list中获取下一个block。

[0059] 优选地,所述系统还包括:

[0060] 删除模块,用于删除ChunkMeta元数据;

[0061] 释放模块,用于当删除所述ChunkMeta元数据成功时,释放block块。

[0062] 优选地,所述释放模块具体用于:

[0063] 将block的id加入空闲block队列free_block_list末尾;

[0064] 将所述block的分配位图block_bit_map的分配位置置0。

[0065] 综上所述,本发明公开了一种追加写场景下顺序存储磁盘管理的方法,当需要对追加写场景下顺序存储的磁盘进行管理时,首先从磁盘加载DiskMeta和ChunkMeta数据信息,然后基于ChunkMeta中起始block_id,按照块加载该chunk所有的block_id到ChunkMeta的有序列表block_list中;扫描ChunkMeta的有序列表block_list,得到空闲的block的队列和block的分配位图block_bit_map;在首次写入数据时,ChunkServer申请chunk,初始化chunk的ChunkMeta信息;在申请chunk的同时,为该chunk申请一个block,作为chunk的起始block;将chunk的数据偏移和大小转成block的内的写入偏移和大小;当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer;当分配block成功,将block的id写入到所述ChunkMeta的有序列表block_list中,同时,将下一个block的id写入前一个block

的末尾数据段，完成落盘操作。本发明能够有效的对裸盘按照块的方式进行管理，灵活定义块的大小，利用追加写的办法提升存储速度。

附图说明

[0066] 为了更清楚地说明本发明实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

[0067] 图1为本发明公开的一种追加写场景下顺序存储磁盘管理的方法实施例1的方法流程图；

[0068] 图2为本发明公开的一种追加写场景下顺序存储磁盘管理的方法实施例2的方法流程图；

[0069] 图3为本发明公开的一种追加写场景下顺序存储磁盘管理的方法实施例3的方法流程图；

[0070] 图4为本发明公开的一种追加写场景下顺序存储磁盘管理的系统实施例1的结构示意图；

[0071] 图5为本发明公开的一种追加写场景下顺序存储磁盘管理的系统实施例2的结构示意图；

[0072] 图6为本发明公开的一种追加写场景下顺序存储磁盘管理的系统实施例3的结构示意图。

具体实施方式

[0073] 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做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。

[0074] 如图1所示，为本发明公开的一种追加写场景下顺序存储磁盘管理的方法实施例1的方法流程图，所述方法可以包括以下步骤：

[0075] S101、将磁盘划分为DiskMeta、ChunkMeta和BlockData；其中，BlockData由固定大小磁盘块组成，通过磁盘物理偏移进行区分；

[0076] 当需要实现对追加写场景下顺序存储的磁盘进行管理时，在本实施例中，磁盘被划分层磁盘元数据段(DiskMeta)，chunk原数据段(ChunkMetas)和BlockData段；其中，BlockData是由固定大小磁盘块组成，通过磁盘物理偏移进行区分。具体的，一个文件由多个chunk组成，一个chunk由多个block组成，chunk中需要存储的BlockData的有序列表(block_list)，每一个BlockData依次使用一个blockId表示。

[0077] 其中，chunk为自定义名称，表示分布式存储系统对外开放提供的最小存储单位；Block为自定义名称，表示分布式存储系统的单机上会将磁盘均匀划分成N个块，每一个块称之为一个Block。BlockData为自定义名称，表示Block的数据块叫做BlockData；DiskMeta为存放磁盘的元数据信息，包括磁盘的大小，磁盘中的Block个数等等。ChunkMeta为存储

chunk的元信息，包括chunk的index，chunk已经使用的长度，可使用的长度，已经分配的磁盘的block的id。

[0078] S102、从磁盘加载DiskMeta和ChunkMeta数据信息；

[0079] 然后对磁盘初始化，按照预先约定的DiskMeta大小、ChunkMeta大小和BlockData大小对磁盘的进行初始化。具体的，在对磁盘进行初始化时，首先从磁盘加载DiskMeta和ChunkMeta数据信息。

[0080] S103、基于ChunkMeta中起始block_id，按照块加载该chunk所有的block_id到ChunkMeta的有序列表block_list中；

[0081] 然后，按照ChunkMeta中的起始block_id，按照块加载该chunk所有的block_id到ChunkMeta的有序列表block_list中。

[0082] S104、扫描ChunkMeta的有序列表block_list，得到空闲的block的队列和block的分配位图block_bit_map；

[0083] 然后，扫描ChunkMeta的有序列表block_list，得到空闲的Block的队列(free_block_list)和block的分配位图。其中，Block分配位图是一个二字节的位图block_bit_map，通过0,1标识block是否被使用。

[0084] S105、在首次写入数据时，ChunkServer申请chunk，初始化chunk的ChunkMeta信息；

[0085] 在数据写入过程中，在首次写入时，通过ChunkServer申请chunk，初始化chunk的ChunkMeta信息。其中，Chunkserver为自定义名称，是一种服务，管理chunk、ChunkMeta、Block的服务，负责接收和处理Client的文件存储请求。

[0086] S106、在申请chunk的同时，为该chunk申请一个block，作为chunk的起始block；

[0087] 然后，在申请chunk的同时，需要为该chunk申请一个block，该block作为chunk的起始block。

[0088] S107、将chunk的数据偏移和大小转成block的内的写入偏移和大小；

[0089] 然后，将chunk的数据偏移和大小转换成block的内的写入偏移和大小。

[0090] S108、当写入数据大小超过一个block的大小，分配一个新的block给ChunkServer；

[0091] 若写入大小超过一个block的大小，ChunkServer会尝试分配一个新的block给ChunkServer。

[0092] S109、当分配block成功，将block的id写入到ChunkMeta的有序列表block_list中，同时，将下一个block的id写入前一个block的末尾数据段，完成落盘操作。

[0093] 若分配block成功，将block的id写入到ChunkMeta的block_list中，同时，将下一个block的id写入前一个block的末尾数据段。之后，完成落盘操作。

[0094] 综上所述，在上述实施例中，当需要对追加写场景下顺序存储的磁盘进行管理时，首先从磁盘加载DiskMeta和ChunkMeta数据信息，然后基于ChunkMeta中起始block_id，按照块加载该chunk所有的block_id到ChunkMeta的有序列表block_list中；扫描ChunkMeta的有序列表block_list，得到空闲的block的队列和block的分配位图block_bit_map；在首次写入数据时，ChunkServer申请chunk，初始化chunk的ChunkMeta信息；在申请chunk的同时，为该chunk申请一个block，作为chunk的起始block；将chunk的数据偏移和大小转成

block的内的写入偏移和大小;当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer;当分配block成功,将block的id写入到所述ChunkMeta的有序列表block_list中,同时,将下一个block的id写入前一个block的末尾数据段,完成落盘操作。本发明能够有效的对裸盘按照块的方式进行管理,灵活定义块的大小,利用追加写的办法提升存储速度。

[0095] 具体的,在上述实施例1中,当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer的其中一种实现方式如图2所示,可以包括以下步骤:

[0096] S201、判断空闲block队列free_block_list是否为空,若是,则进入S202,若否,则进入S203:

[0097] 在分配新block

[0098] S202、返回磁盘满的状态码;

[0099] S203、从空闲block队列free_block_list中获取第一个block;

[0100] S204、通过block的分配位图block_bit_map判断该block是否被使用,若否,则进入S205:

[0101] S205、该block分配成功,将block的分配位图block_bit_map的分配的位置置1。

[0102] 具体的,在上述实施例1的基础上还可以进一步实现对数据的处理,如图3所示,可以包括以下步骤:

[0103] S301、按照chunk上的偏移计算读取数据在chunk的第几个block上;

[0104] S302、基于计算结果,在ChunkMeta中的有序列表block_list中,获取block的id;

[0105] S303、按照block的id计算磁盘上的偏移地址,按照偏移读取数据;

[0106] S304、判断读取数据大小是否超过当前的block,若否,则进入S305,若是,则进入S306:

[0107] S305、读取结束返回数据;

[0108] S306、从ChunkMeta中的有序列表block_list中获取下一个block;

[0109] S307、删除ChunkMeta元数据;

[0110] S308、当删除ChunkMeta元数据成功时,释放block块。

[0111] 具体的,可以通过将block的id加入空闲block队列free_block_list末尾,将block的分配位图block_bit_map的分配位置置0,实现block块的释放。

[0112] 需要说明的是,本发明提供的追加写场景下顺序存储磁盘管理的方法,能够适用的磁盘存储介质包括但不限于SSD、HDD盘等存储介质。

[0113] 综上所述,本发明将下一个block的id写入上一个block的数据末尾,减少了ChunkMeta持久化操作的次数。同时,能够在ChunkMeta元数据丢失的情况下,将数据存储的位置进行恢复。通过block_bit_map和free_block_list管理空闲的block,能够完成block的快速的分配和回收,简化了块管理的上下文信息,减少缓存管理的空间。block的大小可以灵活定义,减小了顺序存储的场景下的块管理的磁盘空间浪费。在追加读写情况下,能够动态分配灵活定义大小的磁盘块提供给用户使用。

[0114] 如图4所示,为本发明公开的一种追加写场景下顺序存储磁盘管理的系统实施例1的结构示意图,所述系统可以包括:

[0115] 定义模块401,用于将磁盘划分为DiskMeta、ChunkMeta和BlockData;其中,所述

BlockData由固定大小磁盘块组成,通过磁盘物理偏移进行区分;

[0116] 当需要实现对追加写场景下顺序存储的磁盘进行管理时,在本实施例中,磁盘被划分层磁盘元数据段(DiskMeta),chunk原数据段(ChunkMetas)和BlockData段;其中,BlockData是由固定大小磁盘块组成,通过磁盘物理偏移进行区分。具体的,一个文件由多个chunk组成,一个chunk由多个block组成,chunk中需要存储的BlockData的有序列表(block_list),每一个BlockData依次使用一个blockId表示。

[0117] 其中,chunk为自定义名称,表示分布式存储系统对外开放提供的最小存储单位;Block为自定义名称,表示分布式存储系统的单机上会将磁盘均匀划分成N个块,每一个块称之为一个Block。BlockData为自定义名称,表示Block的数据块叫做BlockData;DiskMeta为存放磁盘的元数据信息,包括磁盘的大小,磁盘中的Block个数等等。ChunkMeta为存储chunk的元信息,包括chunk的index,chunk已经使用的长度,可使用的长度,已经分配的磁盘的block的id。

[0118] 第一加载模块402,用于从磁盘加载DiskMeta和ChunkMeta数据信息;

[0119] 然后对磁盘初始化,按照预先约定的DiskMeta大小、ChunkMeta大小和BlockData大小对磁盘的进行初始化。具体的,在对磁盘进行初始化时,首先从磁盘加载DiskMeta和ChunkMeta数据信息。

[0120] 第二加载模块403,用于基于ChunkMeta中起始block_id,按照块加载该chunk所有的block_id到ChunkMeta的有序列表block_list中;

[0121] 然后,按照ChunkMeta中的起始block_id,按照块加载该chunk所有的block_id到ChunkMeta的有序列表block_list中。

[0122] 扫描模块404,用于扫描ChunkMeta的有序列表block_list,得到空闲的block的队列和block的分配位图block_bit_map;

[0123] 然后,扫描ChunkMeta的有序列表block_list,得到空闲的Block的队列(free_block_list)和block的分配位图。其中,Block分配位图是一个二字节的位图block_bit_map,通过0,1标识block是否被使用。

[0124] 初始化模块405,用于在首次写入数据时,ChunkServer申请chunk,初始化chunk的ChunkMeta信息;

[0125] 在数据写入过程中,在首次写入时,通过ChunkServer申请chunk,初始化chunk的ChunkMeta信息。其中,Chunkserver为自定义名称,是一种服务,管理chunk、ChunkMeta、Block的服务,负责接收和处理Client的文件存储请求。

[0126] 申请模块406,用于在申请chunk的同时,为该chunk申请一个block,作为chunk的起始block;

[0127] 然后,在申请chunk的同时,需要为该chunk申请一个block,该block作为chunk的起始block。

[0128] 转换模块407,用于将chunk的数据偏移和大小转成block的内的写入偏移和大小;

[0129] 然后,将chunk的数据偏移和大小转换成block的内的写入偏移和大小。

[0130] 分配模块408,用于当写入数据大小超过一个block的大小,分配一个新的block给ChunkServer;

[0131] 若写入大小超过一个block的大小,ChunkServer会尝试分配一个新的block给

ChunkServer。

[0132] 写入模块409，用于当分配block成功，将block的id写入到所述ChunkMeta的有序列表block_list中，同时，将下一个block的id写入前一个block的末尾数据段，完成落盘操作。

[0133] 若分配block成功，将block的id写入到ChunkMeta的block_list中，同时，将下一个block的id写入前一个block的末尾数据段。之后，完成落盘操作。

[0134] 综上所述，在上述实施例中，当需要对追加写场景下顺序存储的磁盘进行管理时，首先从磁盘加载DiskMeta和ChunkMeta数据信息，然后基于ChunkMeta中起始block_id，按照块加载该chunk所有的block_id到ChunkMeta的有序列表block_list中；扫描ChunkMeta的有序列表block_list，得到空闲的block的队列和block的分配位图block_bit_map；在首次写入数据时，ChunkServer申请chunk，初始化chunk的ChunkMeta信息；在申请chunk的同时，为该chunk申请一个block，作为chunk的起始block；将chunk的数据偏移和大小转成block的内的写入偏移和大小；当写入数据大小超过一个block的大小，分配一个新的block给ChunkServer；当分配block成功，将block的id写入到所述ChunkMeta的有序列表block_list中，同时，将下一个block的id写入前一个block的末尾数据段，完成落盘操作。本发明能够有效的对裸盘按照块的方式进行管理，灵活定义块的大小，利用追加写的办法提升存储速度。

[0135] 具体的，在上述系统实施例1中，分配模块在执行当写入数据大小超过一个block的大小，分配一个新的block给ChunkServer时的其中一种实现方式如图5所示，可以包括：

[0136] 第一判断单元501，用于判断空闲block队列free_block_list是否为空；

[0137] 返回单元502，用于当空闲block队列free_block_list为空时，返回磁盘满的状态码；

[0138] 获取单元503，用于当空闲block队列free_block_list不为空时，从空闲block队列free_block_list中获取第一个block；

[0139] 第二判断单元504，用于通过block的分配位图block_bit_map判断该block是否被使用；

[0140] 设置单元505，用于当通过所述block的分配位图block_bit_map判断该block未被使用时，该block分配成功，将block的分配位图block_bit_map的分配的位置置1。

[0141] 具体的，在上述系统实施例1的基础上还可以进一步实现对数据的处理，如图6所示，可以包括：

[0142] 计算模块601，用于按照chunk上的偏移计算读取数据在chunk的第几个block上；

[0143] 第一获取模块602，用于基于计算结果，在ChunkMeta中的有序列表block_list中，获取block的id；

[0144] 读取模块603，用于按照block的id计算磁盘上的偏移地址，按照偏移读取数据；

[0145] 判断模块604，用于判断读取数据大小是否超过当前的block；

[0146] 返回模块605，用于当读取数据大小未超过是否当前的block时，读取结束返回数据；

[0147] 第二获取模块606，用于当读取数据大小未超过当前的block时，从ChunkMeta中的有序列表block_list中获取下一个block；

[0148] 删除模块607,用于删除ChunkMeta元数据;

[0149] 释放模块608,用于当删除ChunkMeta元数据成功时,释放block块。

[0150] 具体的,释放模块可以通过将block的id加入空闲block队列free_block_list末尾,将block的分配位图block_bit_map的分配位置置0,实现block块的释放。

[0151] 需要说明的是,本发明提供的追加写场景下顺序存储磁盘管理的方法,能够适用的磁盘存储介质包括但不限于SSD、HDD盘等存储介质。

[0152] 综上所述,本发明将下一个block的id写入上一个block的数据末尾,减少了ChunkMeta持久化操作的次数。同时,能够在ChunkMeta元数据丢失的情况下,将数据存储的位置进行恢复。通过block_bit_map和free_block_list管理空闲的block,能够完成block的快速的分配和回收,简化了块管理的上下文信息,减少缓存管理的空间。block的大小可以灵活定义,减小了顺序存储的场景下的块管理的磁盘空间浪费。在追加读写情况下,能够动态分配灵活定义大小的磁盘块提供给用户使用。

[0153] 本说明书中各个实施例采用递进的方式描述,每个实施例重点说明的都是与其他实施例的不同之处,各个实施例之间相同相似部分互相参见即可。对于实施例公开的装置而言,由于其与实施例公开的方法相对应,所以描述的比较简单,相关之处参见方法部分说明即可。

[0154] 专业人员还可以进一步意识到,结合本文中所公开的实施例描述的各示例的单元及算法步骤,能够以电子硬件、计算机软件或者二者的结合来实现,为了清楚地说明硬件和软件的可互换性,在上述说明中已经按照功能一般性地描述了各示例的组成及步骤。这些功能究竟以硬件还是软件方式来执行,取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能,但是这种实现不应认为超出本发明的范围。

[0155] 结合本文中所公开的实施例描述的方法或算法的步骤可以直接用硬件、处理器执行的软件模块,或者二者的结合来实施。软件模块可以置于随机存储器(RAM)、内存、只读存储器(ROM)、电可编程ROM、电可擦除可编程ROM、寄存器、硬盘、可移动磁盘、CD-ROM、或技术领域内所公知的任意其它形式的存储介质中。

[0156] 对所公开的实施例的上述说明,使本领域专业技术人员能够实现或使用本发明。对这些实施例的多种修改对本领域的专业技术人员来说将是显而易见的,本文中所定义的一般原理可以在不脱离本发明的精神或范围的情况下,在其它实施例中实现。因此,本发明将不会被限制于本文所示的这些实施例,而是要符合与本文所公开的原理和新颖特点相一致的最宽的范围。

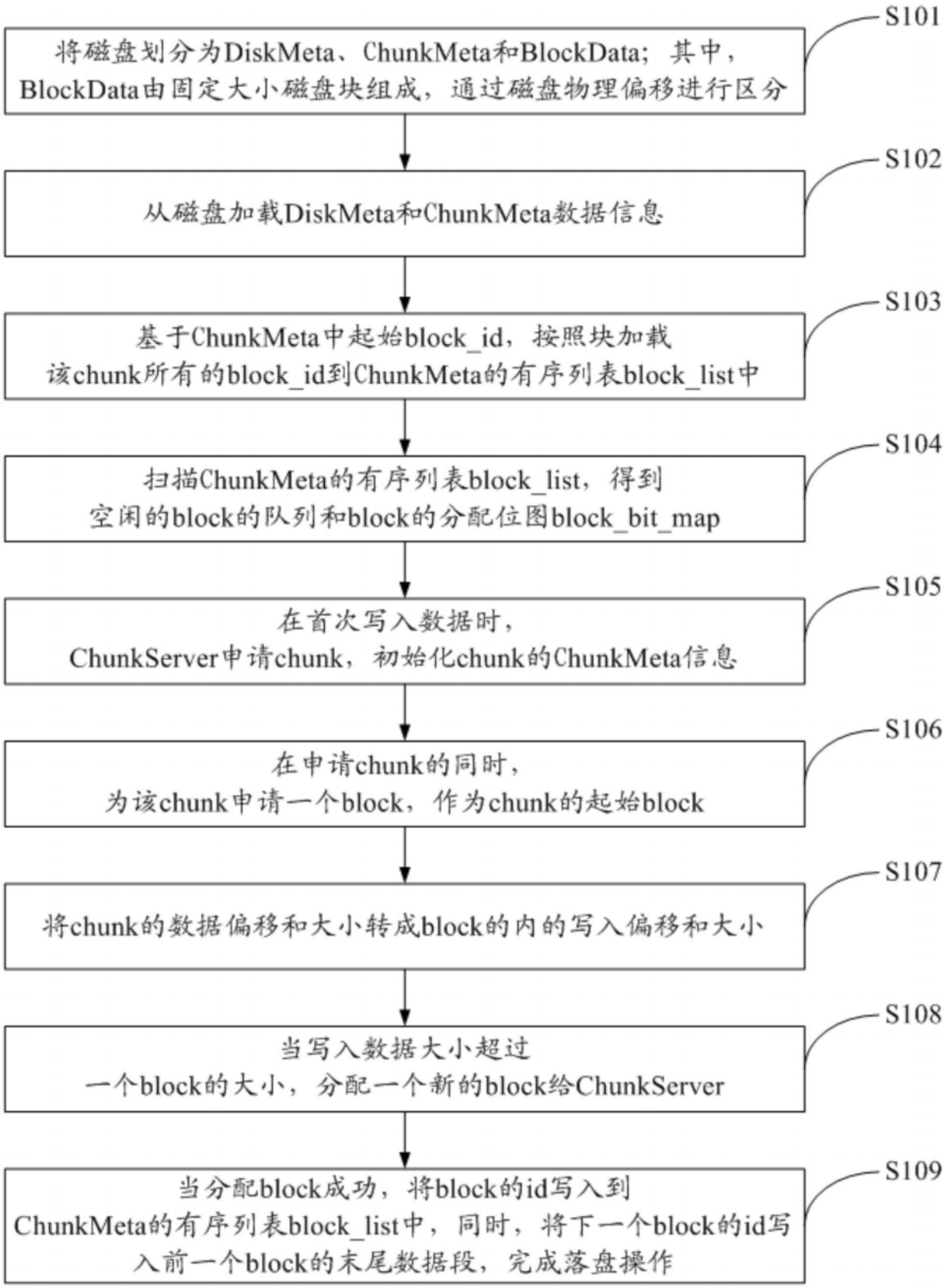

S101
将磁盘划分为DiskMeta、ChunkMeta和BlockData；其中，BlockData由固定大小磁盘块组成，通过磁盘物理偏移进行区分
S102
从磁盘加载DiskMeta和ChunkMeta数据信息
S103
基于ChunkMeta中起始block_id，按照块加载该chunk所有的block_id到ChunkMeta的有序列表block_list中
S104
扫描ChunkMeta的有序列表block_list，得到空闲的block的队列和block的分配位图block_bit_map
S105
在首次写入数据时，ChunkServer申请chunk，初始化chunk的ChunkMeta信息
S106
在申请chunk的同时，为该chunk申请一个block，作为chunk的起始block
S107
将chunk的数据偏移和大小转成block的内的写入偏移和大小
S108
当写入数据大小超过一个block的大小，分配一个新的block给ChunkServer
S109
当分配block成功，将block的id写入到ChunkMeta的有序列表block_list中，同时，将下一个block的id写入前一个block的末尾数据段，完成落盘操作

图1

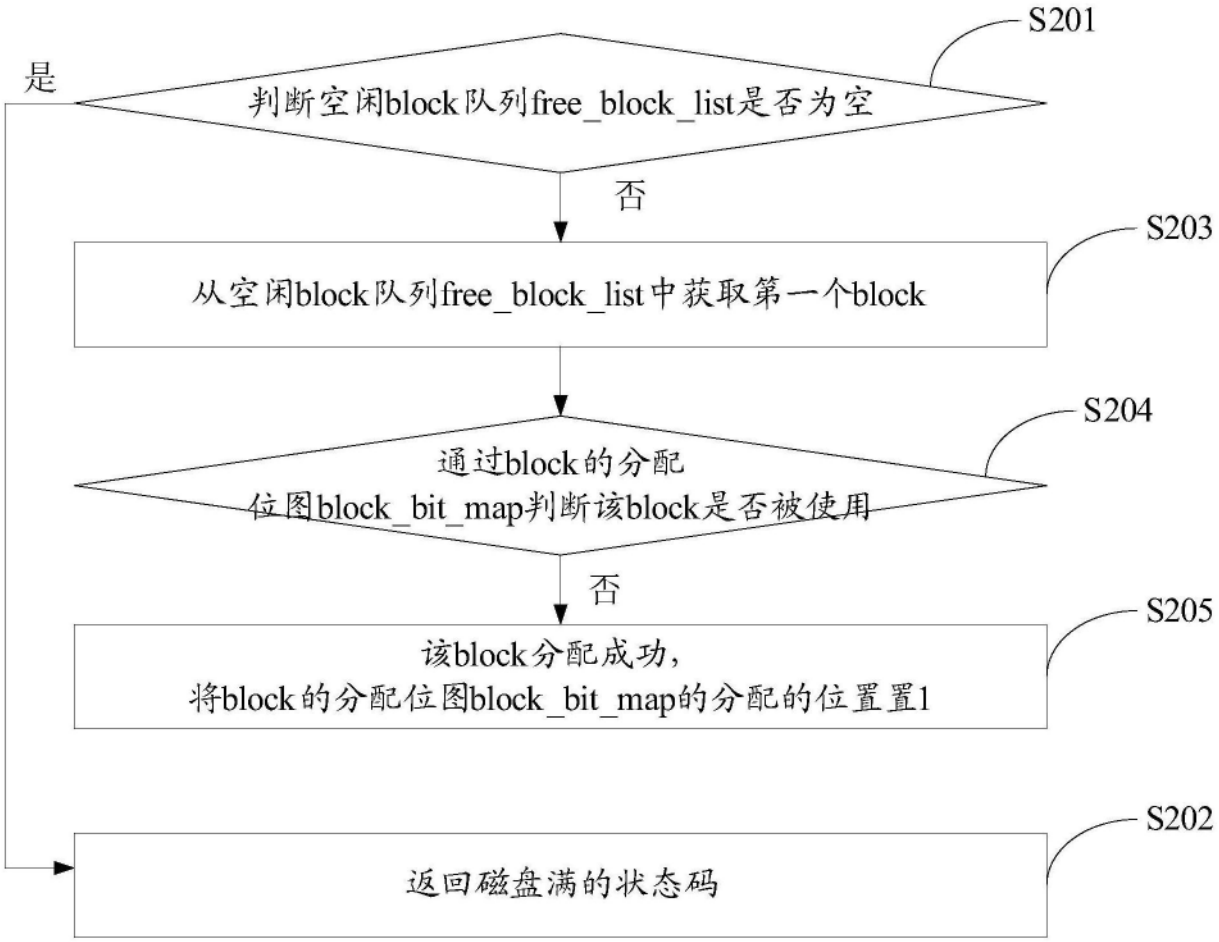

S201
判断空闲block队列free_block_list是否为空
是
否
S203
从空闲block队列free_block_list中获取第一个block
S204
通过block的分配
位图block_bit_map判断该block是否被使用
否
S205
该block分配成功，
将block的分配位图block_bit_map的分配的位置置1
S202
返回磁盘满的状态码

图2

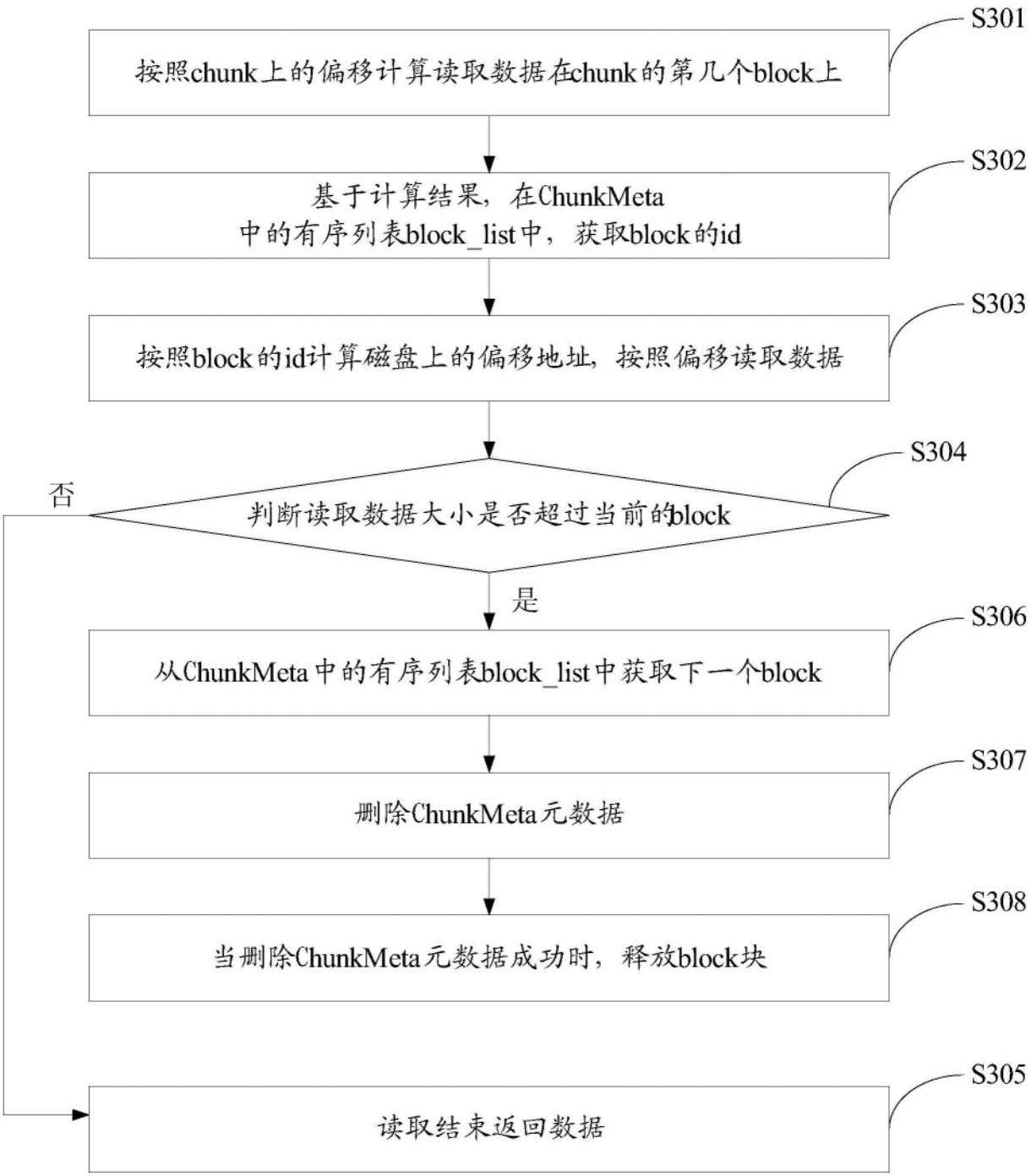

S301
按照chunk上的偏移计算读取数据在chunk的第几个block上
S302
基于计算结果，在ChunkMeta
中的有序列表block_list中，获取block的id
S303
按照block的id计算磁盘上的偏移地址，按照偏移读取数据
S304
判断读取数据大小是否超过当前的block
否
是
S306
从ChunkMeta中的有序列表block_list中获取下一个block
S307
删除ChunkMeta元数据
S308
当删除ChunkMeta元数据成功时，释放block块
S305
读取结束返回数据

图3

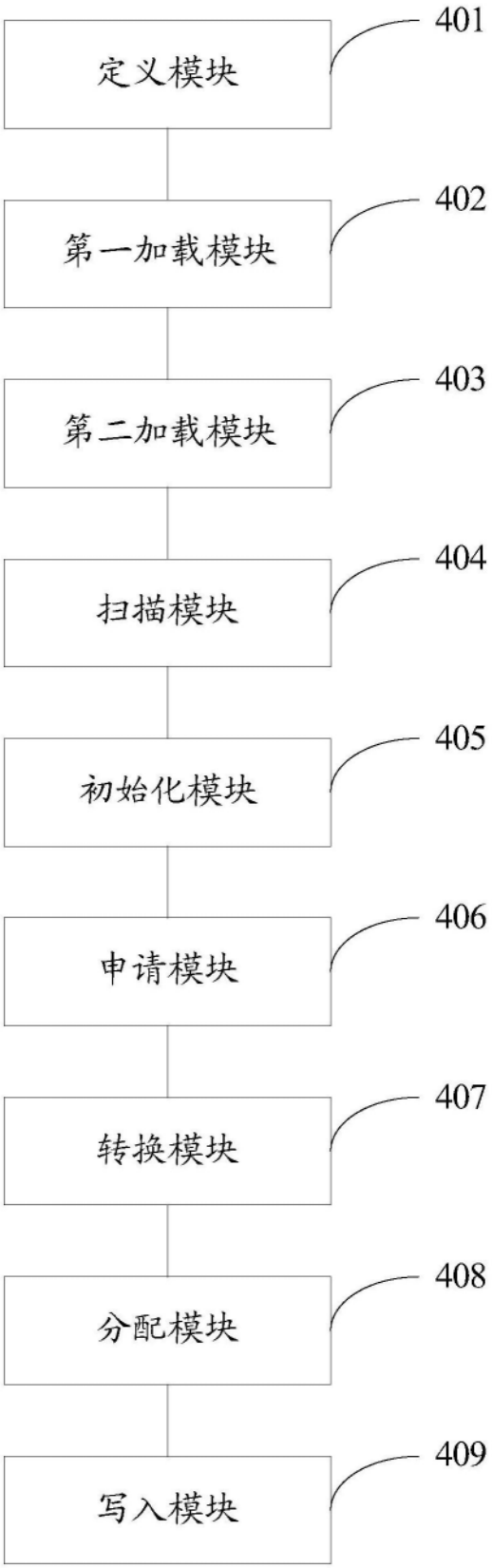

401
定义模块
402
第一加载模块
403
第二加载模块
404
扫描模块
405
初始化模块
406
申请模块
407
转换模块
408
分配模块
409
写入模块

图4

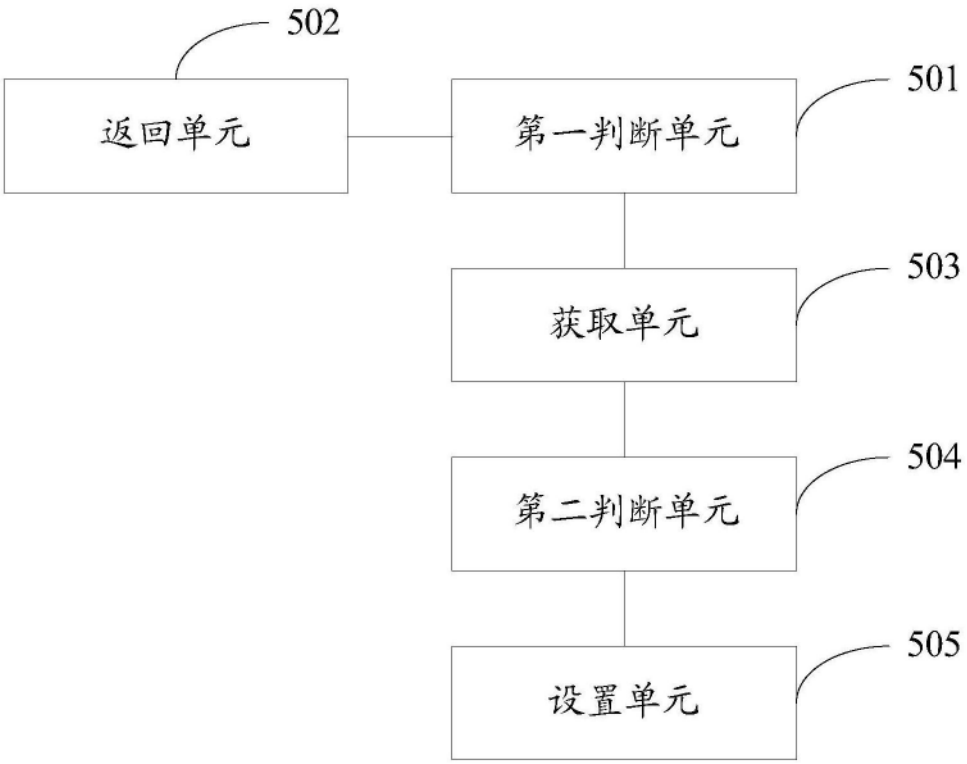


图5

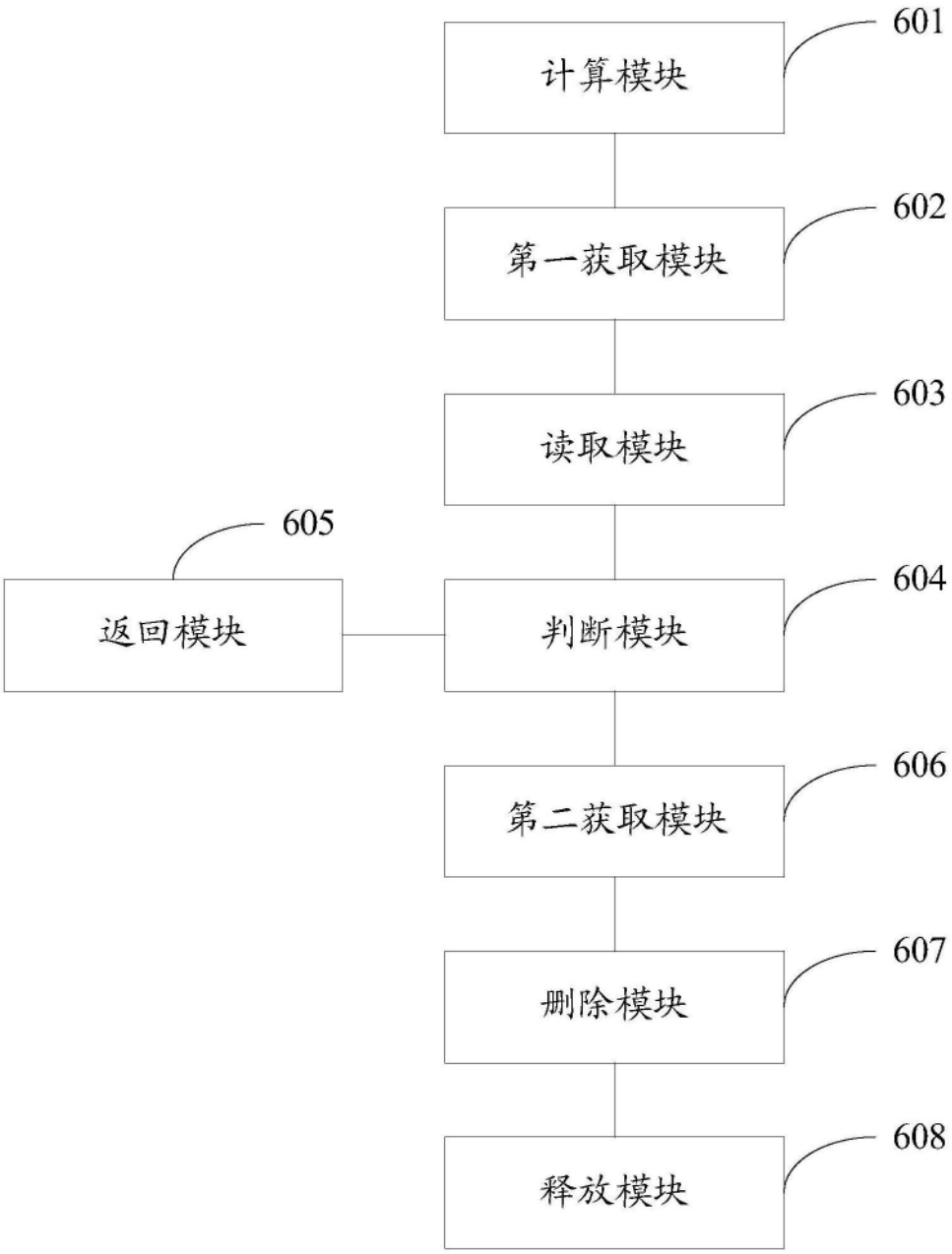

601
计算模块
602
第一获取模块
603
读取模块
605
返回模块
604
判断模块
606
第二获取模块
607
删除模块
608
释放模块

图6