(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 119166344 A
(43)申请公布日 2024.12.20

(21)申请号 202411226589.7

(22)申请日 2024.09.03

(71)申请人 深圳前海微众银行股份有限公司
地址 518000 广东省深圳市南山区沙河西路1819号深圳湾科技生态园7栋A座

(72)发明人 郑奕彬 彭荣杰 王顺云 王文虎 侯银花

(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 赵蕊

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 11/14(2006.01)

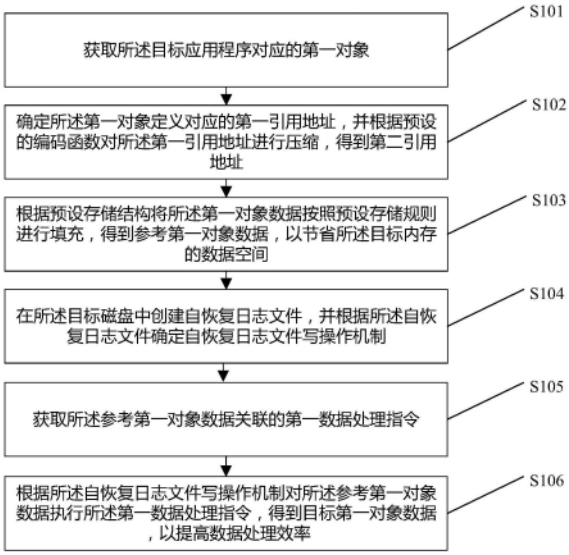
权利要求书3页 说明书21页 附图7页

(54)发明名称

针对内存受限的数据处理方法及相关装置

(57)摘要

本申请提供了一种针对内存受限的数据处理方法及相关装置,所述方法包括:获取所述目标应用程序对应的第一对象;确定所述第一对象定义对应的第一引用地址,并根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址;根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据;在所述目标磁盘中创建自恢复日志文件,并根据所述自恢复日志文件确定自恢复日志文件写操作机制;获取所述参考第一对象数据关联的第一数据处理指令;根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令,得到目标第一对象数据,以提高数据处理效率。



CN 119166344 A

1.一种针对内存受限的数据处理方法,其特征在于,应用于数据解析转换机,目标设备包括所述数据解析转换机、目标应用程序、目标内存和目标磁盘,包括:

获取所述目标应用程序对应的第一对象;所述第一对象存储在所述目标内存;所述第一对象包括第一对象数据和第一对象定义;

确定所述第一对象定义对应的第一引用地址,并根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址;所述第二引用地址用于指向所述第一对象定义;

根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据,以节省所述目标内存的数据空间;

在所述目标磁盘中创建自恢复日志文件,并根据所述自恢复日志文件确定自恢复日志文件写操作机制;所述自恢复日志文件写操作机制用于存储数据处理操作,以确保数据在系统崩溃或故障后能够恢复;

获取所述参考第一对象数据关联的第一数据处理指令;

根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令,得到目标第一对象数据,以提高数据处理效率。

2.如权利要求1所述的方法,其特征在于,所述根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址,包括:

获取所述第一引用地址对应的内存区域的起始偏移量;

将所述第一引用地址与所述起始偏移量进行相减,得到参考引用地址;

将所述参考引用地址无符号右移a位以进行地址压缩,得到所述第二引用地址;a为正整数。

3.如权利要求1所述的方法,其特征在于,所述第一对象数据包括多个子数据;所述预设存储结构包括多层数据空间,每一层数据空间的内存大小相同;所述根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据,包括:

若所述第一对象数据无继承层次,则将所述多个子数据按照预设的第一排序规则进行排序,得到第一子数据序列;

将所述第一子数据序列按照先后顺序填充至所述多层数据空间,得到所述参考第一对象数据;

若所述第一对象数据包括多个继承层次,则将所述多个继承层次中每一继承层次的子数据按照所述第一排序规则进行排序,得到多个参考第二子数据序列;每一继承层次对应一个参考第二子数据序列;

根据继承层次从上到下的顺序对所述多个参考第二子数据序列进行排序,得到目标第二子数据序列;

将所述目标第二子数据序列按照先后顺序填充至所述多层数据空间,得到所述参考第一对象数据。

4.如权利要求3所述的方法,其特征在于,所述方法还包括:

若所述第一对象数据包括多个继承层次,则获取继承层次i的至少一个子数据;所述继承层次i为所述多个继承层次中任意一个继承层次;

若所述至少一个子数据对应的数据类型中存在预设第一类型,且继承层次i-1对应的数据空间未满,则将所述至少一个子数据按照预设的第二排序规则进行排序,得到参考第

三子数据序列;

将所述参考第三子数据序列按照先后顺序填充至所述继承层次i-1对应的数据空间;

当所述继承层次i-1对应的数据空间无法存入数据时,将未存入数据空间的所有子数据按照所述第一排序规则进行排序,得到参考第四子数据序列;

将所述参考第四子数据序列按照先后顺序填充至所述多层数据空间,得到所述参考第一对象数据。

5.如权利要求1所述的方法,其特征在于,所述自恢复日志文件包括写入指针和检查指针,所述写入指针用于指示逆时针写入数据的起始位置,所述检查指针用于指示所述自恢复日志文件中数据的起始位置,所述在所述目标磁盘中创建自恢复日志文件,并根据所述自恢复日志文件确定自恢复日志文件写操作机制,包括:

在所述目标磁盘中获取k个日志文件;所述k个日志文件中每一日志文件大小一致;k为正整数;

将所述k个日志文件按照预设的环形结构进行整合,得到所述自恢复日志文件;

当所述自恢复日志文件执行参考磁盘顺序写操作时,对所述写入指针和所述检查指针的位置进行检查;

当所述检查指针在所述写入指针之后,则确定所述自恢复日志文件中数据未满,继续执行所述参考磁盘顺序写操作;

当所述检查指针与所述写入指针重合,则确定所述自恢复日志文件中数据已满,获取所述检查指针的第一位置;

根据预设移动距离控制所述检查指针逆时针向前移动,得到所述检查指针的第二位置;

确定所述第一位置和所述第二位置之间的数据为待处理数据,并对所述待处理数据进行处理,得到空余的参考日志文件空间;

根据所述参考日志文件空间继续执行所述参考磁盘顺序写操作。

6.如权利要求5所述的方法,其特征在于,所述自恢复日志文件包括第一自恢复日志文件,所述根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令,得到目标第一对象数据,包括:

根据所述参考第一对象数据确定第一原始数据页;

根据所述第一数据处理指令对所述第一原始数据页中的所述参考第一对象数据进行修改,得到第一数据变更页;所述第一数据变更页包括变更第一对象数据;

通过对所述第一自恢复日志文件执行第一磁盘顺序写操作,将所述第一数据处理指令存储至所述第一自恢复日志文件;

当针对所述第一自恢复日志文件的读写操作满足预设第一条件时,通过对所述第一数据变更页执行第一磁盘随机写操作,将所述变更第一对象数据写入所述目标磁盘,得到所述目标第一对象数据。

7.如权利要求1-6任一项所述的方法,其特征在于,所述自恢复日志文件还包括第二自恢复日志文件,所述方法还包括:

获取所述目标应用程序对应的第二对象;所述第二对象存储在所述目标磁盘;

确定所述第二对象对应的第二对象数据关联的第二数据处理指令;所述第二数据处理

指令包括至少一个数据操作；

在所述目标内存中创建数据处理缓存页；

将所述第二数据处理指令存储至所述数据处理缓存页，并对所述第二数据处理指令中至少一个数据操作进行整合，得到目标数据处理指令；

通过对所述第二自恢复日志文件执行第二磁盘顺序写操作，将所述目标数据处理指令存储至所述第二自恢复日志文件。

8.如权利要求7所述的方法，其特征在于，在所述将所述目标数据处理指令存储至所述第二自恢复日志文件之后，所述方法还包括：

通过对所述目标磁盘执行第二磁盘随机写操作，将所述第二对象数据从所述目标磁盘加载到所述目标内存，得到第二原始数据页；

根据所述目标数据处理指令对所述第二原始数据页中的所述第二对象数据进行处理，得到第二数据变更页；所述第二数据变更页包括变更第二对象数据；

通过对所述第二自恢复日志文件执行第三磁盘顺序写操作，将所述目标数据处理指令存储至所述第二自恢复日志文件；

当针对所述第二自恢复日志文件的读写操作满足预设第二条件时，通过对所述第二数据变更页执行第三磁盘随机写操作，将所述变更第二对象数据写入所述目标磁盘，得到目标第二对象数据。

9.一种针对内存受限的数据处理装置，其特征在于，应用于数据解析转换机，目标设备包括所述数据解析转换机、目标应用程序、目标内存和目标磁盘，所述装置包括获取模块、确定模块、填充模块、创建模块以及执行模块，其中：

所述获取模块，用于获取所述目标应用程序对应的第一对象；所述第一对象存储在所述目标内存；所述第一对象包括第一对象数据和第一对象定义；

所述确定模块，用于确定所述第一对象定义对应的第一引用地址，并根据预设的编码函数对所述第一引用地址进行压缩，得到第二引用地址；所述第二引用地址用于指向所述第一对象定义；

所述填充模块，用于根据预设存储结构将所述第一对象数据按照预设存储规则进行填充，得到参考第一对象数据，以节省所述目标内存的数据空间；

所述创建模块，用于在所述目标磁盘中创建自恢复日志文件，并根据所述自恢复日志文件确定自恢复日志文件写操作机制；所述自恢复日志文件写操作机制用于存储数据处理操作，以确保数据在系统崩溃或故障后能够恢复；

所述获取模块，还用于获取所述参考第一对象数据关联的第一数据处理指令；

所述执行模块，用于根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令，得到目标第一对象数据，以提高数据处理效率。

10.一种电子设备，其特征在于，包括：处理器、存储器、通信接口以及一个或多个程序；所述一个或多个程序被存储在所述存储器中，并且被配置成由所述处理器执行，所述程序包括用于执行如权利要求1-8任一项所述的方法中的步骤的指令。

针对内存受限的数据处理方法及相关装置

技术领域

[0001] 本申请涉及数据存储技术领域,尤其涉及一种针对内存受限的数据处理方法及相关装置。

背景技术

[0002] 针对大量数据进行计算时,若机器资源充足,则可以通过多台机器构建的分布式存储和分布式计算集群来完成计算,而只有单台机器时,在计算过程中则会出现内存不足的问题。目前,为解决这个问题,一般是通过页面置换算法、内存压缩算法、交换空间机制对内存进行压缩和交换,从而在一定程度上节省和扩大内存空间。

[0003] 但是,目前方案中的页面置换算法并没有直接优化内存数据的计算,而是基于页面的访问频率或最近使用情况做出决策,可能会发生频繁的页面置换,从而导致大量的磁盘的读写操作,从而降低系统的整体性能。其中,内存压缩算法也只有被选定压缩的数据才能享受到压缩带来的内存空间节省,无法满足全部内存数据。另外,磁盘的读写操作将会消耗大量的计算资源,从而导致计算资源集中在读写操作上,降低了机器的处理能力。

[0004] 因此,如何对内存数据进行存储优化,并改进内存计算和文件交互机制,以优化内存利用率并提高大量数据计算的处理性能,成为待解决的问题。

发明内容

[0005] 本申请实施例提供一种针对内存受限的数据处理方法及相关装置,实现了对内存数据的存储优化,并对内存计算和文件交互机制进行改进,以优化内存利用率并提高大量数据计算的处理性能。

[0006] 第一方面,本申请实施例提供了一种针对内存受限的数据处理方法,应用于数据解析转换机,目标设备包括所述数据解析转换机、目标应用程序、目标内存和目标磁盘,所述方法包括:

[0007] 获取所述目标应用程序对应的第一对象;所述第一对象存储在所述目标内存;所述第一对象包括第一对象数据和第一对象定义;

[0008] 确定所述第一对象定义对应的第一引用地址,并根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址;所述第二引用地址用于指向所述第一对象定义;

[0009] 根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据,以节省所述目标内存的数据空间;

[0010] 在所述目标磁盘中创建自恢复日志文件,并根据所述自恢复日志文件确定自恢复日志文件写操作机制;所述自恢复日志文件写操作机制用于存储数据处理操作,以确保数据在系统崩溃或故障后能够恢复;

[0011] 获取所述参考第一对象数据关联的第一数据处理指令;

[0012] 根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令,得到目标第一对象数据,以提高数据处理效率。

[0013] 第二方面,本申请实施例提供了一种针对内存受限的数据处理装置,应用于数据解析转换机,目标设备包括所述数据解析转换机、目标应用程序、目标内存和目标磁盘,所述装置包括获取模块、确定模块、填充模块、创建模块以及执行模块,其中:

[0014] 所述获取模块,用于获取所述目标应用程序对应的第一对象;所述第一对象存储在所述目标内存;所述第一对象包括第一对象数据和第一对象定义;

[0015] 所述确定模块,用于确定所述第一对象定义对应的第一引用地址,并根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址;所述第二引用地址用于指向所述第一对象定义;

[0016] 所述填充模块,用于根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据,以节省所述目标内存的数据空间;

[0017] 所述创建模块,用于在所述目标磁盘中创建自恢复日志文件,并根据所述自恢复日志文件确定自恢复日志文件写操作机制;所述自恢复日志文件写操作机制用于存储数据处理操作,以确保数据在系统崩溃或故障后能够恢复;

[0018] 所述获取模块,还用于获取所述参考第一对象数据关联的第一数据处理指令;

[0019] 所述执行模块,用于根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令,得到目标第一对象数据,以提高数据处理效率。

[0020] 第三方面,本申请实施例提供一种电子设备,包括处理器、存储器、通信接口以及一个或多个程序,其中,上述一个或多个程序被存储在上述存储器中,并且被配置由上述处理器执行,上述程序包括用于执行本申请实施例第一方面任一方法中的步骤的指令。

[0021] 第四方面,本申请实施例提供了一种计算机可读存储介质,其中,上述计算机可读存储介质存储用于电子数据交换的计算机程序,其中,上述计算机程序使得计算机执行如本申请实施例第一方面任一方法中所描述的部分或全部步骤。

[0022] 第五方面,本申请实施例提供了一种计算机程序产品,其中,上述计算机程序产品包括存储了计算机程序的非瞬时性计算机可读存储介质,上述计算机程序可操作来使计算机执行如本申请实施例第一方面任一方法中所描述的部分或全部步骤。该计算机程序产品可以为一个软件安装包。

[0023] 通过实施本申请实施例,实现了对内存数据的存储优化,并对内存计算和文件交互机制进行改进,以优化内存利用率并提高大量数据计算的处理性能。

附图说明

[0024] 为了更清楚地说明本申请实施例技术方案,下面将对实施例描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图是本申请的一些实施例,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。

[0025] 图1是本申请实施例提供的一种针对内存受限的数据处理方法的流程示意图;

[0026] 图2是本申请实施例提供的一种第一优化机制的示意图;

[0027] 图3是本申请实施例提供的一种对象数据的第一内存结构示意图;

[0028] 图4是本申请实施例提供的一种对象数据的第二内存结构示意图;

[0029] 图5是本申请实施例提供的一种OBJ1的内存结构示意图;

[0030] 图6是本申请实施例提供的一种OBJ2的内存结构示意图;

[0031] 图7是本申请实施例提供的一种自恢复日志文件的结构示意图;

[0032] 图8是本申请实施例提供的一种第二优化机制的示意图;

[0033] 图9是本申请实施例提供的一种电子设备的结构示意图;

[0034] 图10是本申请实施例提供的一种针对内存受限的数据处理装置的功能模块组成框图。

具体实施方式

[0035] 为了使本技术领域的人员更好地理解本申请方案,下面将结合本申请实施例中的附图,对本申请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申请一部分实施例,而不是全部的实施例。基于本申请中的实施例,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,都属于本申请保护的范围。

[0036] 本申请的说明书和权利要求书及上述附图中的术语“第一”、“第二”等是用于区别不同对象,而不是用于描述特定顺序。此外,术语“包括”和“具有”以及它们任何变形,意图在于覆盖不排他的包含。例如包含了一系列步骤或单元的过程、方法、系统、产品或设备没有限定于已列出的步骤或单元,而是可选地还包括没有列出的步骤或单元,或可选地还包括对于这些过程、方法、产品或设备固有的其他步骤或单元。

[0037] 应理解,本文中术语“和/或”,仅仅是一种描述关联对象的关联关系,表示可以存在三种关系,例如,A和/或B,可以表示:单独存在A,同时存在A和B,单独存在B这三种情况。另外,本文中字符“/”,表示前后关联对象是一种“或”的关系。本申请实施例中出现的“多个”是指两个或两个以上。

[0038] 本申请实施例中的“至少一项(个)”或其类似表达,是指这些项中的任意组合,包括单项(个)或复数项(个)的任意组合,是指一个或多个,多个指的是两个或两个以上。例如,a、b或c中的至少一项(个),可以表示如下七种情况:a,b,c,a和b,a和c,b和c,a、b和c。其中,a、b、c中的每一个可以是元素,也可以是包含一个或多个元素的集合。

[0039] 本申请实施例中出现的“连接”是指直接连接或者间接连接等各种连接方式,以实现设备间的通信,本申请实施例对此不做任何限定。

[0040] 在本文中提及“实施例”意味着,结合实施例描述的特定特征、结构或特性可以包含在本申请的至少一个实施例中。在说明书中的各个位置出现该短语并不一定均是指相同的实施例,也不是与其它实施例互斥的独立的或备选的实施例。本领域技术人员显式地和隐式地理解的是,本文所描述的实施例可以与其它实施例相结合。

[0041] 下面先对本申请涉及到的相关名词进行解释,解释如下:

[0042] 对象数据:int、char这些是基本数据类型,而对象数据则是包含多种基本数据类型的一种复杂数据结构。

[0043] 磁盘IO:是指计算机系统中对磁盘设备进行的数据读写操作。I为数据输入,即读取操作,O为数据输出,即写入操作。

[0044] 磁盘随机读写IO:在读写数据时,需要磁盘先做寻道定位到数据点,且每次读写数据都需要进行定位,这种IO操作就是磁盘随机读写IO。

[0045] 磁盘顺序读写IO:当磁头已经定位到指定数据点,且需要执行后续的读写操作,则只需要从指定数据点继续往后读而无需重新磁头寻道,则这种IO操作就是磁盘顺序读写

IO。

[0046] 页面表:记录内存数据页面的情况。

[0047] 引用地址:指向某个内存单元起始地址的编号。

[0048] 中央处理器(Central Processing Unit,CPU):计算机系统中的主要处理器,负责执行指令、处理数据和控制计算机的各种操作。它是计算机的核心组件之一,通常被称为“计算机的大脑”。

[0049] merge操作:特指将“数据处理缓存页”中的数据合并更新到“原始数据页”的动作。

[0050] purge操作:特指“原始数据页“”有数据变化后,形成“数据变更页”,并在合适的时机将“数据变更页”的数据写入到目标磁盘的动作。

[0051] 针对大量数据进行计算时,若机器资源充足,则可以通过多台机器构建的分布式存储和分布式计算集群来完成计算,而只有单台机器时,在计算过程中则会出现内存不足的问题。目前,为解决这个问题,一般是通过页面置换算法、内存压缩算法、交换空间机制对内存进行压缩和交换,从而在一定程度上节省和扩大内存空间。但是,目前方案中的页面置换算法并没有直接优化内存数据的计算,而是基于页面的访问频率或最近使用情况做出决策,可能会发生频繁的页面置换,从而导致大量的磁盘的读写操作,从而降低系统的整体性能。其中,内存压缩算法也只有被选定压缩的数据才能享受到压缩带来的内存空间节省,无法满足全部内存数据。另外,磁盘的读写操作将会消耗大量的计算资源,从而导致计算资源集中在读写操作上,降低了机器的处理能力。

[0052] 因此,如何对内存数据进行存储优化,并改进内存计算和文件交互机制,以优化内存利用率并提高大量数据计算的处理性能,成为待解决的问题。

[0053] 为解决上述问题,本申请实施例提供了一种针对内存受限的数据处理方法及相关装置,应用于数据解析转换机,目标设备包括所述数据解析转换机、目标应用程序、目标内存和目标磁盘,首先,获取所述目标应用程序对应的第一对象;所述第一对象存储在所述目标内存;所述第一对象包括第一对象数据和第一对象定义;然后,确定所述第一对象定义对应的第一引用地址,并根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址;所述第二引用地址用于指向所述第一对象定义;再根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据,以节省所述目标内存的数据空间;在所述目标磁盘中创建自恢复日志文件,并根据所述自恢复日志文件确定自恢复日志文件写操作机制;所述自恢复日志文件写操作机制用于存储数据处理操作,以确保数据在系统崩溃或故障后能够恢复;最后,获取所述参考第一对象数据关联的第一数据处理指令;根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令,得到目标第一对象数据,以提高数据处理效率。实现了对内存数据的存储优化,并对内存计算和文件交互机制进行改进,以优化内存利用率并提高大量数据计算的处理性能。

[0054] 下面结合图1对本申请实施例中的一种针对内存受限的数据处理方法进行说明,图1为本申请实施例提供的一种针对内存受限的数据处理方法的流程示意图,应用于数据解析转换机,目标设备包括所述数据解析转换机、目标应用程序、目标内存和目标磁盘,具体包括以下步骤:

[0055] 步骤S101,获取所述目标应用程序对应的第一对象。

[0056] 其中,所述第一对象存储在所述目标内存;所述第一对象包括第一对象数据和第

一对象定义。

[0057] 具体的,目标设备包括但不限于计算机、智能手机、平板电脑、游戏机、服务器,在此不做具体限定。通过数据解析转换机对目标内存中的内存缓存页进行读取,并根据内存缓存页对应的内存起始地址,访问目标应用程序对应的第一对象。数据解析转换机用于处理和转换数据,可以对数据进行数据解析、数据转换、数据集成、数据清洗及预处理,以确保数据在不同系统和应用程序之间的一致性和准确性。

[0058] 步骤S102,确定所述第一对象定义对应的第一引用地址,并根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址。

[0059] 其中,所述第二引用地址用于指向所述第一对象定义。

[0060] 其中,根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址,具体包括:

[0061] 获取所述第一引用地址对应的内存区域的起始偏移量;将所述第一引用地址与所述起始偏移量进行相减,得到参考引用地址;将所述参考引用地址无符号右移a位以进行地址压缩,得到所述第二引用地址;a为正整数。

[0062] 具体的,当目标应用程序执行写入数据操作时,数据解析转换机先判断该数据对应的类型是否为引用地址。若该数据为引用地址,则调用预设的编码函数对数据进行压缩,该编码函数可以是encode_oop(ref_address)函数,在此不做具体限定。然后,将压缩后的数据存储到内存区域。若该数据为非引用地址,则无需进行压缩,直接存储到内存区域。其中,在调用第一编码函数对数据进行压缩的过程中,可以获取第一引用地址对应的内存区域的起始偏移量,然后,将第一引用地址与起始偏移量进行相减,得到参考引用地址,再将参考引用地址无符号右移a位以进行地址压缩,得到第二引用地址。需要说明的是,该引用地址压缩操作的算法逻辑较为简单,且与现有的内存压缩算法不同,该压缩操作消耗的算力资源可忽略不计。

[0063] 在一个可能的实施例中,当目标应用程序执行读取数据操作时,数据解析转换机先判断该数据对应的类型是否为引用地址。如果该数据为引用地址,则调用预设的解码函数进行解压,该编码函数可以是decode_oop(narrow_address)函数,在此不做具体限定。然后,对解压后的引用地址进行内存寻址,访问到实际数据所在位置后再返回给目标应用程序。如果该数据为非引用地址,则无需进行解压,直接根据该数据指向的内存起始地址进行内存寻址,访问到实际数据后返回给目标应用程序。其中,在调用预设的解码函数进行解压的过程中,可以将压缩后的第二引用地址无符号左移a位,再加上起始偏移量,得到初始的第一引用地址。

[0064] 可见,通过压缩引用地址,可以显著减少内存的使用量,从而降低内存带宽的需求,对目标设备的整体性能有积极影响。

[0065] 便于理解,参见图2,图2为申请实施例提供的一种第一优化机制的示意图,可见,目标内存中去除了原有的引用地址映射表,新增了数据解析转换机,并将对象数据区域指向对象定义区域。其中,目标应用程序指向目标内存中的页面表,在该页面表中,“原始”表示其对应的内存数据页为目标内存中的原始数据;“压缩”表示其对应的内存数据页为目标内存中的原始数据经过压缩后,得到的数据;“磁盘”表示其对应的内存数据页为目标内存中的原始数据替换到目标磁盘后,得到的数据。通过数据解析转换机,指向内存数据页对应

的对象数据区域,再由对象数据区域指向对应定义区域。其中Pg为内存数据页标识,例如,Pg1对应内存数据页1。OBJ为对象标识,则OBJ1和OBJ2分别为第一对象和第二对象,则OBJ1数据为第一对象数据,OBJ1定义为第一对象定义,OBJ2数据为第二对象数据,OBJ2定义为第二对象定义。

[0066] 需要说明的是,原有的引用地址映射表中存储了每个对象对应的两个引用地址,即对象定义引用地址和对象数据引用地址,若在64位操作系统下,该引用地址将会各自占用8B的存储空间,从而导致每个对象都会占用16B的存储空间。为节省该部分的内存存储,可以通过数据解析转换机,将目标应用程序指向对象数据区域,即原有的引用地址映射表中对象数据引用地址所指向的内存区域,以节省8B的内存存储。然后,将对象数据的内存结构进行改造,即在对象数据头部增加一个对象定义引用地址,以代替原有的引用地址映射表中对象定义引用地址。

[0067] 便于理解,参见图3,图3为申请实施例提供的一种对象数据的第一内存结构示意图,可见,OBJ1对应的对象定义引用地址的变量名为ref_md1(OBJ1),OBJ2对应的对象定义引用地址的变量名为ref_md2(OBJ2),OBJ2中指向OBJ1的引用地址的变量名为ref_obj1,即OBJ2的对象数据中存储了指向OBJ1的对象数据的引用地址ref_obj1。需要说明的是,灰色区域用于表示内存填充,实际为空白内存数据,无实际数据含义,每一行区域均表示8B的数据空间。其中,可以通过数据解析转换机来优化所有引用地址的内存消耗,以便节省内存存储。

[0068] 需要说明的是,在原先的内存访问方式中,基本都是目标应用程序读取内存对象的引用地址,然后再进行相关的内存操作。在64位操作系统下,CPU数据总线一次性都是读取8B的数据,无论是写入数据到内存,还是从内存读取数据,如果采用这种处理机制,那么引用地址占用内存空间就是8B,而非引用地址则没有这个限制,可以是1B,也可以是2B等。为了尽量避免CPU数据总线跨缓存行访问数据,一般会通过内存对齐和内存填充来保证数据的内存空间是8B对齐,以提高数据处理效率。例如,如果将要读取的完整数据分散在两个缓存行中,那么需要读取两次才能把数据凑齐。所以,将8B的引用地址进行压缩优化,不仅要考虑数据的对齐需要,还要考虑压缩后内存的寻址能力是否不够用等问题,比如把8B压缩成5B、6B、7B,则内存的寻址能力是够用的,其对应的寻址能力分别为$2^8\times4GB$、$2^{16}\times4GB$、$2^{24}\times4GB$,但是这种压缩方式,将会导致内存的对齐和填充较为困难。

[0069] 可知,对象数据的8种基本数据类型包括long、double、float、int、short、byte、boolean、char,其中,有2种是占用8B的(例如,long、double),有2种是占用4B的(例如,float、int),还有1种是占用2B的(例如,short),最后3种是占用1B的(例如,byte、boolean、char)。当8B压缩成5B时,还剩下3B的空间可能需要内存空白填充,也可以用short、byte、boolean、char这4种数据来填充;当8B压缩成6B时,还剩下2B可能需要内存空白填充,也可以用short、byte、boolean、char这4种数据来填充;当8B压缩成7B时,还剩下1B可能需要内存空白填充,也可以用byte、boolean、char这3种数据来填充。而如果把8B压缩成4B,其内存寻址能力还可以保持为4GB,且能用来对齐的基本数据类型是6种(例如,float、int、short、byte、boolean、char)。由此可知,压缩成4B可以尽可能节省内存的对齐和填充带来的额外内存消耗。

[0070] 另外,在改造后的内存结构中,对象数据区域头部包括一个8B的引用地址,再加上

对象数据区域中的数据,对象的内存占用空间大小总是大于8B,也就是说对象的内存起始地址均为8的倍数(包含0),即0x0000、0x0008、0x0010、0x0018、0x0020。可知,在二进制表达中,该内存起始地址的最低3位的二进制码总是“000”。因此,可将该3个比特位用于优化内存寻址,即实际内存的寻址能力是$2^3 \times 4GB = 32GB$。可知,将引用地址8B压缩成4B,不仅解决了内存寻址的需要,还解决了压缩引用地址内存消耗的问题。

[0071] 便于理解,参见图4,图4为申请实施例提供的一种对象数据的第二内存结构示意图,可见,不仅对象数据定义引用地址可以进行压缩,指向其他对象的引用地址也可以进行压缩,例如,OBJ2中指向OBJ1的引用地址变量ref_obj1,从8B压缩为4B的数据大小。

[0072] 需要说明的是,在对引用地址进行压缩后,其数据的读取和写入无法依赖原来的内存机制,因为操作系统默认是按8B存储,也是按8B进行读取的,但在压缩后,对数据是按4B进行存储的。因此,可以通过数据解析转换机对数据进行处理,即判断数据是否为引用地址,并根据不同的数据操作,对数据进行压缩或者解压。其中,将8B压缩为4B时,需要执行无符号右移32位操作,而为了增加内存寻址能力,又考虑到对象数据占用内存空间大小以8B对齐,故可以减少右移3位,即只需要无符号右移29位即可。由此可知,a可以为29,用于表示无符号右移29位以执行压缩操作。

[0073] 步骤S103,根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据,以节省所述目标内存的数据空间。

[0074] 其中,所述第一对象数据包括多个子数据;所述预设存储结构包括多层数据空间,每一层数据空间的内存大小相同;所述根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据,具体包括:

[0075] 若所述第一对象数据无继承层次,则将所述多个子数据按照预设的第一排序规则进行排序,得到第一子数据序列;将所述第一子数据序列按照先后顺序填充至所述多层数据空间,得到所述参考第一对象数据;

[0076] 若所述第一对象数据包括多个继承层次,则将所述多个继承层次中每一继承层次的子数据按照所述第一排序规则进行排序,得到多个参考第二子数据序列;每一继承层次对应一个参考第二子数据序列;根据继承层次从上到下的顺序对所述多个参考第二子数据序列进行排序,得到目标第二子数据序列;将所述目标第二子数据序列按照先后顺序填充至所述多层数据空间,得到所述参考第一对象数据。

[0077] 需要说明的是,第一对象数据可以是OBJ1对象数据,则该OBJ1对象数据可以包括多个子数据,如char c1、int i1、int i12等。其中,预设存储结构包括多层数据空间,每一层数据空间的内存大小均为8B。

[0078] 在一个可能的实施例中,若所述第一对象数据包括多个继承层次,则获取继承层次i的至少一个子数据;所述继承层次i为所述多个继承层次中任意一个继承层次;若所述至少一个子数据对应的数据类型中存在预设第一类型,且继承层次i-1对应的数据空间未满,则将所述至少一个子数据按照预设的第二排序规则进行排序,得到参考第三子数据序列;将所述参考第三子数据序列按照先后顺序填充至所述继承层次i-1对应的数据空间;当所述继承层次i-1对应的数据空间无法存入数据时,将未存入数据空间的所有子数据按照所述第一排序规则进行排序,得到参考第四子数据序列;将所述参考第四子数据序列按照先后顺序填充至所述多层数据空间,得到所述参考第一对象数据。

[0079] 需要说明的是,虽然引用地址通过压缩操作从8B变成4B,但是因为内存对齐填充的原因,实际上节省的4B的数据空间还是需要空白数据进行补齐。因此,可以根据预设存储结构将对象数据按照预设存储规则进行填充,以节省目标内存的数据空间。

[0080] 其中,预设存储规则如下:

[0081] 规则1、对象均以8B进行对齐。

[0082] 规则2、对象数据区域中的对象数据按照预设的第一排序规则进行对齐,且第一排序规则为将对象数据按照如下顺序进行排序:double=long>float=int>short>byte>char>boolean>ref,其中,ref表示引用地址。

[0083] 规则3、若对象数据包括多个继承层次,则每个继承层次对应的数据区域不要混合排序,可以按照继承层次从上到下进行排序,上面的继承层次排序完成后,再对下面的继承层次进行排序。其中,同一继承层次中的数据按照第一排序规则进行排序。

[0084] 规则4、若对象数据包括多个继承层次,当前继承层次中的第一个子数据的数据类型为预设第一类型,且上一继承层次未用完8B的数据空间,则数据解析转换机破坏所述规则2,将子数据按照预设的第二排序规则进行排序,并对齐填充到剩余的数据空间中。等到该数据空间按照8B对齐后,对剩余的数据重新按照所述规则2继续排序。其中,预设第一类型为double或者long,第二排序规则为将对象数据按照如下顺序排序:int>short>byte>ref。

[0085] 规则5、若对象数据中的第一个子数据的数据类型是预设第一类型,且对象定义引用地址还没用完8B的数据空间,则如所述规则4相同处理。

[0086] 需要说明的是,对象数据存储按照新的规则进行存储,也按照新的规则进行读取,如引用地址压缩和解压的操作相同,可以通过数据解析转换机实现。

[0087] 在一个可能的实施例中,在OBJ1的数据结构中,其数据区域不存在double或long,所以,虽然对象头部的对象定义引用地址只使用了4B,未使用完8B,但无法满足规则5的全部条件,所以不可用,且OBJ1没有继承层次,所以也无法使用规则3和规则4进行优化,那么就通过规则1和规则2进行优化调整即可。

[0088] 其中,原始的OBJ1的数据结构如下:

```
    OBJ1:
{char c1; //1B
 int i1; //4B
```

[0089]

```
 int i12; //4B
short s1; //2B
 float f1; //4B
byte t1; //1B}
```

[0090] 将OBJ1按照预设存储规则进行调整,得到调整后的OBJ1如下:

```
OBJ1:
{ float f1; //4B
  int i1; //4B
  int i12; //4B
 short s1; //2B
  byte t1; //1B
 char c1; //1B}
```

[0092] 需要说明的是,OBJ1(i1=1,s1=1)表示OBJ1的一个对象数据,其中,int数据类型的变量i1,即i1=1表示该对象数据的标识为1,short数据类型的变量s1,即s1=1表示该对象数据的数值为1。

[0093] 便于理解,参见图5,图5为申请实施例提供的一种OBJ1的内存结构示意图,可见,ref_md1占用第一层数据空间中的4B,剩余4B通过空白数据进行填充,f1和i1分别占用第二层数据空间的4B,i12占用第三层数据空间的2B,t1和c1分别占用第三层数据空间的1B,则总占用内存大小为24B。

[0094] 可知,OBJ1原来的数据区域占用内存是24B,加上引用地址映射表的16B,总共是占用内存空间40B,而现在去除了引用地址映射表,且所有数据都位于数据区域,总占用内存大小为24B。因此,相对于原来的数据存储方式,其内存占用大小从40B减少为24B,即节省了40%的内存空间。

[0095] 在一个可能的实施例中,OBJ2的数据包括两层继承层次,再加上OBJ2的自身的数据区域,可构成三层继承层次的复杂对象。在第一继承层次中,因为存在double类型,且对象头部的对象定义引用地址未用完8B,即符合规则5,可以参照规则4打破原有的规则2排序规则,把“int ib1”先排序到对象定义引用地址变量“ref_md2”的后面,然后再排序“doubledb1”,此时第一继承层次已经排序完毕,且用完了8B。

[0096] 在第二继承层次中,只有变量“float fb2”,按照规则2和规则3正常排序即可,此时第二层继承层次没有用完8B。在第三继承层次中,第一个子数据的数据类型为long,且第二层继承层次没有用完8B,也就是满足了规则3的同时,也满足了规则4的条件,那么优先按照规则3进行排序,即把“int i2”提到第二继承层次中“float fb2”的后面,这样一来第二继承层次也就用完了8B,所以此时重新恢复规则2的排序规则,故接下来排序的数据是“long l2”,然后是“short s2”,接着是“byte b2”,最后是“ref_OBJ1 ref_obj1”。

[0097] 其中,原始的OBJ1的数据结构如下:

```
OBJ2 extends OBJ_BASE2:
    {double db1; //8B
     int ib1; //4B
     ----------
    float fb2; //4B
[0098]  int i2; //4B
     ----------
     long l2; //8B
    short s2; //2B
     byte b2; //1B
  ref_OBJ1 ref_obj1; }
```

[0099] 其中,虚线用于区分不同的继承层次,无实际意义,且OBJ2包括OBJ_BASE2的数据。

[0100] 将OBJ2按照预设存储规则进行调整,得到调整后的OBJ2如下:

```
OBJ2 extends OBJ_BASE2:
     {int ib1; //4B
      ----------
    double db1; //8B
      ----------
    float fb2; //4B
[0101]  int i2; //4B
      ----------
     long l2; //8B
    short s2; //2B
     byte b2; //1B
  ref_OBJ1 ref_obj1; }
```

[0102] 便于理解,参见图6,图6为申请实施例提供的一种OBJ2的内存结构示意图,可见,OBJ2原来的数据区域占用内存是56B,加上引用地址映射表的16B,总共是占用内存空间72B,而现在没有了引用地址映射表,且所有数据都位于数据区域,总占用内存大小根据图6可算出为40B。因此,相对于原来的数据存储方式,其内存占用大小从72B减少为40B,即节省了44.45%的内存空间。

[0103] 可知,利用数据解析转换机对所有对象数据进行事前压缩,即使得目标应用程序指向对象数据的内存数据区域,以节省对象数据引用地址所占用的内存,再在对象数据头部增加对象定义引用地址,并对其进行压缩。然后,根据预设存储规则,优化对象数据存储结构,以解决因空白数据补齐导致内存消耗过多的问题,节省内存空间。

[0104] 步骤S104,在所述目标磁盘中创建自恢复日志文件,并根据所述自恢复日志文件确定自恢复日志文件写操作机制。

[0105] 其中,所述自恢复日志文件写操作机制用于存储数据处理操作,以确保数据在系统崩溃或故障后能够恢复。

[0106] 其中,所述自恢复日志文件包括写入指针和检查指针,所述写入指针用于指示逆时针写入数据的起始位置,所述检查指针用于指示所述自恢复日志文件中数据的起始位置,所述在所述目标磁盘中创建自恢复日志文件,并根据所述自恢复日志文件确定自恢复日志文件写操作机制,具体包括:

[0107] 在所述目标磁盘中获取k个日志文件;所述k个日志文件中每一日志文件大小一致;k为正整数;将所述k个日志文件按照预设的环形结构进行整合,得到所述自恢复日志文件;当所述自恢复日志文件执行参考磁盘顺序写操作时,对所述写入指针和所述检查指针的位置进行检查;当所述检查指针在所述写入指针之后,则确定所述自恢复日志文件中数据未满,继续执行所述参考磁盘顺序写操作;当所述检查指针与所述写入指针重合,则确定所述自恢复日志文件中数据已满,获取所述检查指针的第一位置;根据预设移动距离控制所述检查指针逆时针向前移动,得到所述检查指针的第二位置;确定所述第一位置和所述第二位置之间的数据为待处理数据,并对所述待处理数据进行处理,得到空余的参考日志文件空间;根据所述参考日志文件空间继续执行所述参考磁盘顺序写操作。

[0108] 便于理解,参见图7,图7为申请实施例提供的一种自恢复日志文件的结构示意图,可见,当k为10时,10个日志文件形成一个环形,其中,日志文件可以为1GB,且每个日志文件的大小相同,在此不做具体限定。其中,10个日志文件依次编号为revylog_file0、revylog_file1、revylog_file2、revylog_file3、……、revylog_file9。write_point表示写入指针,可以按逆时针的顺序写入文件,而check_point表示检查指针,也可以按逆时针的顺序进行检查数据。该日志文件在逻辑上首尾相连,形成一个环形,上一个文件写完了,则继续写下一个,然后循环从头开始,可以在逻辑实现上支持循环追加写的操作,以便得到自恢复日志文件写操作机制。

[0109] 其中,revylog_file0衔接着revylog_file9,当写入指针落后于检查指针时,表示当前10个revylog日志文件还没全部写完,而写入指针落与检查指针之间的内容就是暂存的数据内容;当写入指针追上检查指针时,表示10个revylog日志文件全部写完了,也就是当前没有内容可以追加写了,需要停下来,先推动检查指针往前走,等处理完部分暂存数据留出足够空间后,再继续推动写入指针写入其他数据。其中,检查指针往前走的位置可以根据目标应用程序对应的系统的繁忙程度确定,例如,当系统较为繁忙时,可以先处理10%的数据,即检查指针可以往前走到下一个revylog日志文件的位置,当系统较为空闲时,可以一次性处理全部的数据,即检查指针可以绕10个revylog日志文件一圈并回到原来的位置。

[0110] 步骤S105,获取所述参考第一对象数据关联的第一数据处理指令。

[0111] 具体的,第一数据处理指令包括对第一对象数据的写操作指令,例如,第一对象数据为i=1,其关联的第一数据处理指令为OBJ1(i1=1,s1+2)、OBJ1(i1=1,s1-1)。

[0112] 步骤S106,根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令,得到目标第一对象数据,以提高数据处理效率。

[0113] 其中,所述自恢复日志文件包括第一自恢复日志文件,所述根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令,得到目标第一对象数据,具体包括:

[0114] 根据所述参考第一对象数据确定第一原始数据页;根据所述第一数据处理指令对所述第一原始数据页中的所述参考第一对象数据进行修改,得到第一数据变更页;所述第

一数据变更页包括变更第一对象数据;通过对所述第一自恢复日志文件执行第一磁盘顺序写操作,将所述第一数据处理指令存储至所述第一自恢复日志文件;当针对所述第一自恢复日志文件的读写操作满足预设第一条件时,通过对所述第一数据变更页执行第一磁盘随机写操作,将所述变更第一对象数据写入所述目标磁盘,得到所述目标第一对象数据。

[0115] 需要说明的时,预设第一条件包括但不限于第一自恢复日志文件的容量不足,在此不做具体限定。

[0116] 在一个可能的实施例中,目标内存中包括i1＝1、i1＝2、i1＝3这3个OBJ1对应的对象数据,目标磁盘中有i1＝1、i1＝2、i1＝3、i1＝4这4个OBJ1对应的对象数据。当访问i1＝1、i1＝2、i1＝3这3个对象数据时,可以直接访问目标内存,无需从目标磁盘加载;而访问i1＝4、i1＝5、i1＝6这3个对象数据时,则需从目标磁盘加载,也就是多了3次磁盘随机读IO,其中,首次访问没有数据也需要从目标磁盘加载后得知。然后,执行目标指令序列,因为不同OBJ1的对象数据对应的指令都是打散的,所以当指令执行后有写操作时,无法用到文件缓冲区做优化,实际上每一条指令都对应一次磁盘随机写IO,所以总共21条指令需要21次磁盘随机写IO,加上i1＝4、i1＝5、i1＝6这3个OBJ1的首次访问加载的磁盘随机读IO,总共是消耗了24次磁盘随机读写IO。

[0117] 其中,目标指令序列如下所示:

[0118] OBJ1(i1＝1,s1+2)

[0119] OBJ1(i1＝3,s1+1)

[0120] OBJ1(i1＝6,s1+2)

[0121] OBJ1(i1＝2,s1+2)

[0122] OBJ1(i1＝1,s1-1)

[0123] OBJ1(i1＝2,s1+3)

[0124] OBJ1(i1＝1,s1+3)

[0125] OBJ1(i1＝2,s1-5)

[0126] OBJ1(i1＝1,s1-1)

[0127] OBJ1(i1＝3,s1+1)

[0128] OBJ1(i1＝5,s1+5)

[0129] OBJ1(i1＝3,s1+1)

[0130] OBJ1(i1＝4,s1+10)

[0131] OBJ1(i1＝3,s1*)

[0132] OBJ1(i1＝5,s1+5)

[0133] OBJ1(i1＝4,s1-1)

[0134] OBJ1(i1＝4,s1-2)

[0135] OBJ1(i1＝6,s1+3)

[0136] OBJ1(i1＝4,s1-3)

[0137] OBJ1(i1＝5,s1-10)

[0138] OBJ1(i1＝6,s1*)

[0139] 需要说明的是,该目标指令序列指的是一系列需要执行的操作或指令,通常这些指令可能会涉及不同对象数据的修改或操作。而该目标指令序列在执行后,产生的写操作

是分散的,即不连续的写入数据,无法有效地将该写操作合并到缓冲区中进行统一处理,从而导致频繁的文件访问操作。因此,可以通过后续的第二优化机制对目标指令序列的执行过程进行优化,以提高数据处理效率。

[0140] 其中,可以用“int i1”变量作为OBJ1数据的区分要素,例如,将i1=1对应的数据与i1=2对应的数据进行区分,并用“short s1”变量作为数据变更项,将s1初始化为1,即可得到OBJ1(i1=1,s1=1)。其中,执行目标指令序列中的OBJ1(i1=1,s1+2)指令时,将i1=1的对象数据s1增加2。如果i1=1表示的对象数据原来不存在,即不存在OBJ1(i1=1,s1=1),则该操作为初始化对象数据,将s1初始化为2。其中,执行目标指令序列中的OBJ1(i1=1,s1-1)指令时,将i1=1的对象数据s1减少1。如果i1=1表示的对象数据原来不存在,即不存在OBJ1(i1=1,s1=1),则该操作为初始化对象数据,并将s1初始化为-1。其中,执行目标指令序列中的OBJ1(i1=3,s1*)指令时,将i1=3的对象数据进行删除。如果i1=3表示的对象数据原来不存在,则可以忽略该操作。

[0141] 在一个可能的实施例中,OBJ1中的对象数据包括i1=1、i1=2、i1=3,这三个对象数据分别在目标内存中的数据区域为“原始数据页”,表示没有经过任何修改的,跟目标磁盘中数据一一匹配的原始数据。其中,“原始数据页”记录的是OBJ1(i1=1,s1=1),按照OBJ1(i1=1,s1+2)、OBJ1(i1=1,s1-1)、OBJ1(i1=1,s1+3)、OBJ1(i1=1,s1-1)的指令序列进行回放,即s1=1+2-1+3-1,最终结果为s1=4,也就是“数据变更页”最终状态记录的是(i1=1,s1=4)。因此,只需要把s1从1变成4这个结果通过一次随机写IO写入到目标磁盘即可。

[0142] 需要说明的是,因为在目标内存的“数据变更页”中暂存了写操作指令的回放结果,如果在“数据变更页”写到目标磁盘前,目标应用程序死机或掉电,是可能存在丢失数据问题的。因此,可以通过自恢复日志文件写操作机制对数据的处理过程进行优化,即在目标内存的“原始数据页”回放OBJ1 i1=1对象数据的4条写操作指令时,不仅会得到“数据变更页”,同时会把这4条写操作指令顺序追加写到日志文件中,因为是顺序追加写,也就是磁盘顺序写IO,不涉及磁头寻道问题,那么这4条指令可以先写到文件缓冲区,然后一次性刷到目标磁盘中,也就是只需要1次磁盘顺序写IO即可以完成。然后当数据解析转换机空闲时或者目标应用程序关闭时,则把“数据变更页”中的数据通过一次磁盘随机写IO写入到目标磁盘。其中,写完后,第一自恢复日志文件中的检查指针会前进相应的距离,空出占用的空间。或者,第一自恢复日志文件的读写操作满足预设第一条件时,即第一自恢复日志文件容量不足时,则会先触发1次磁盘顺序写IO,这个操作称为purge操作。

[0143] 可知,新的自恢复日志文件写操作机制不是实时通过磁盘随机写IO将写操作指令写入到目标磁盘,而是通过在目标内存“原始数据页”不断回放写操作指令,从而在目标内存中形成“数据变更页”,同时通过文件缓冲区机制优化为1次磁盘顺序IO把这写操作指令序列追加写入到第一自恢复日志文件中,最后在预设条件下,即目标应用程序关闭、系统空闲、第一自恢复日志文件空间不足,再把“数据变更页”的内容通过一次磁盘随机写IO写入到目标磁盘的数据文件中。其中,i1=1的对象数据对比原来4次磁盘随机写IO,新的机制只需要1次磁盘顺序写IO和1次磁盘随机写IO即可。

[0144] 便于理解,参见图8,图8为申请实施例提供的一种第二优化机制的示意图,可见,该第二优化机制包括自恢复日志文件写操作机制,其中,在目标磁盘文件中,revylog用于

表示自恢复日志文件,用于顺序写入内存数据的变更状态,并在启动的时候基于自恢复日志文件恢复内存数据,避免数据丢失,datafile用于表示目标磁盘中的数据持久化磁盘文件。

[0145] 在一个可能的实施例中,i1＝2的OBJ1对象数据,其“原始数据页”记录的是OBJ1(i1＝2,s1＝1),按照OBJ1(i1＝2,s1+2)、OBJ1(i1＝2,s1+3)、OBJ1(i1＝2,s1-5)的指令序列进行回放,即s1＝1+2+3-5,最终结果s1＝1。其中,目标内存的“原始数据页”的最终结果没有发生变化,无需做磁盘随机写IO,但是在回放指令序列的过程中,并不知道最终结果是无变化,所以还是会把该指令序列追加写入到第一自恢复日志文件中,避免意外数据丢失。所以,该过程只需要一次磁盘顺序写IO即可。其中,i1＝2的对象数据对比原来3次磁盘随机写IO,新的机制只需要1次磁盘顺序写IO即可。

[0146] 在一个可能的实施例中,i1＝3的OBJ1对象数据,其“原始数据页”记录的是OBJ1(i1＝3,s1＝1),按照OBJ1(i1＝3,s1+1)、OBJ1(i1＝3,s1+1)、OBJ1(i1＝3,s1+1)、OBJ1(i1＝3,s1*)的指令序列进行回放,即s1＝1+1+1+1*,因为最后一个写操作指令是*,其表示删除这条数据。所以,只需要一次磁盘顺序写IO记录到第一自恢复日志文件,同时需要一次磁盘随机写IO把删除操作写入到磁盘数据中。其中,i1＝3的对象数据对比原来4次磁盘随机写IO,新的机制只需要1次磁盘顺序写IO和1次磁盘随机写IO即可。

[0147] 在一个可能的实施例中,所述自恢复日志文件还包括第二自恢复日志文件,所述方法还具体包括:

[0148] 获取所述目标应用程序对应的第二对象;所述第二对象存储在所述目标磁盘;确定所述第二对象对应的第二对象数据关联的第二数据处理指令;所述第二数据处理指令包括至少一个数据操作;在所述目标内存中创建数据处理缓存页;将所述第二数据处理指令存储至所述数据处理缓存页,并对所述第二数据处理指令中至少一个数据操作进行整合,得到目标数据处理指令;通过对所述第二自恢复日志文件执行第二磁盘顺序写操作,将所述目标数据处理指令存储至所述第二自恢复日志文件。

[0149] 其中,在所述将所述目标数据处理指令存储至所述第二自恢复日志文件之后,所述方法还具体包括:

[0150] 通过对所述目标磁盘执行第二磁盘随机写操作,将所述第二对象数据从所述目标磁盘加载到所述目标内存,得到第二原始数据页;根据所述目标数据处理指令对所述第二原始数据页中的所述第二对象数据进行处理,得到第二数据变更页;所述第二数据变更页包括变更第二对象数据;通过对所述第二自恢复日志文件执行第三磁盘顺序写操作,将所述目标数据处理指令存储至所述第二自恢复日志文件;当针对所述第二自恢复日志文件的读写操作满足预设第二条件时,通过对所述第二数据变更页执行第三磁盘随机写操作,将所述变更第二对象数据写入所述目标磁盘,得到目标第二对象数据。

[0151] 需要说明的是,预设第二条件包括但不限于第二自恢复日志文件的容量不足,在此不做具体限定。

[0152] 在一个可能的实施例中,OBJ1包括i1＝4、i1＝5、i1＝6这三个对象数据,且该三个对象数据均不在内存中。按照原来的机制,在写操作指令执行前,需要先分别通过1次磁盘随机读IO加载到内存中,再回放写操作指令到内存的“原始数据页”。为了优化该问题,可以在目标内存中创建“数据处理缓存页”,即当数据解析转换机判断目标内存中不存在该对象

数据时,可以创建“数据处理缓存页”,然后在该“数据处理缓存页”记录这个写操作指令变更,然后不断继续执行其他写操作指令。其中,可以另外创建一套独立的第二自恢复日志文件来保证异常情况时数据不丢失。

[0153] 其中,“数据处理缓存页”会暂存对象数据对应的写操作指令,不会立马更新到目标磁盘数据中,也不会更新到目标内存的“原始数据页”中,当后续有其他读操作需要访问该对象数据时,数据解析转换机则会从目标磁盘加载该数据到目标内存的“原始数据页”中,并将“数据处理缓存页”就会主动合并到“原始数据页”,该操作称为merge操作,从而形成“数据变更页”,再借助原来的“数据变更页”的purge操作就可以把数据变更内容写入到目标磁盘数据中。如果写操作指令写入到“数据处理缓存页”后,后续一直没有访问该OBJ1数据,或者目标应用程序关闭时,则数据解析转换机主动从目标磁盘加载数据到目标内存,并执行merge操作。所以,“数据处理缓存页”对应的机制也支持启动自恢复。

[0154] 在一个可能的实施例中,i1=4的OBJ1对象数据,在“数据处理缓存页”中,按照OBJ1(i1=4,s1+10)、OBJ1(i1=4,s1-1)、OBJ1(i1=4,s1-2)、OBJ1(i1=4,s1-3)的指令序列进行回放,即s1=10-1-2-3,最终结果s1=4,说明可以通过1次随机写IO写入到目标磁盘即可。其中,i1=4的对象数据对比原来4次磁盘随机写IO,新的机制只需要1次磁盘顺序写IO和1次磁盘随机写IO即可。

[0155] 在一个可能的实施例中,i1=5的OBJ1对象数据,在“数据处理缓存页”中,按照OBJ1(i1=5,s1+5)、OBJ1(i1=5,s1+5)、OBJ1(i1=5,s1-10)的指令序列进行回放,即s1=5+5-10,最终结果s1=0,说明没有数据任何变化。由于,目标磁盘中没有该对象数据,所以无需做磁盘随机写IO。其中,i1=5的对象数据对比原来3次磁盘随机写IO,新的机制只需要1次磁盘顺序写IO即可。

[0156] 在一个可能的实施例中,i1=6的OBJ1对象数据,在“数据处理缓存页”中,按照OBJ1(i1=6,s1+2)、OBJ1(i1=6,s1+3)、OBJ1(i1=6,s1*)的指令序列进行回放,即s1=2+3*,因为最后一个步骤是删除该数据,且目标磁盘中也没有该对象数据,所以无需做磁盘随机写IO。其中,i1=6的对象数据对比原来3次磁盘随机写IO,新的机制只需要1次磁盘顺序写IO即可。

[0157] 可知,对于OBJ1来说,新的机制总共只需要6次磁盘顺序读写IO和3次磁盘随机读写IO,而原来的机制需要24次磁盘随机读写IO。其中,若一次磁盘顺序读写IO耗时是3ms,一次磁盘随机读写IO耗时是10ms,则新旧机制耗时对比如下:

[0158] 新机制总耗时:6×3ms+3×10ms=48ms。

[0159] 旧机制总耗时:24×10ms=240ms。

[0160] 因此,两者的性能差异较为显著,新的机制可以将磁盘随机读写IO转化为磁盘顺序读写IO,并大大减少磁盘IO的次数,从而大大提高大量数据计算的处理性能。

[0161] 下面结合图9对本申请实施例中的电子设备进行说明,图9为本申请实施例提供的一种电子设备的结构示意图,如图9所示,该电子设备包括一个或多个处理器、存储器、通信接口以及一个或多个程序,该处理器通过内部通信总线与该存储器、该通信接口通信连接。

[0162] 其中,处理器主要用于:

[0163] 获取目标应用程序对应的第一对象;第一对象存储在目标内存;第一对象包括第一对象数据和第一对象定义;

[0164] 确定第一对象定义对应的第一引用地址,并根据预设的编码函数对第一引用地址进行压缩,得到第二引用地址;第二引用地址用于指向第一对象定义;

[0165] 根据预设存储结构将第一对象数据按照预设存储规则进行填充,得到参考第一对象数据,以节省目标内存的数据空间;

[0166] 在目标磁盘中创建自恢复日志文件,并根据自恢复日志文件确定自恢复日志文件写操作机制;自恢复日志文件写操作机制用于存储数据处理操作,以确保数据在系统崩溃或故障后能够恢复;

[0167] 获取参考第一对象数据关联的第一数据处理指令;

[0168] 根据自恢复日志文件写操作机制对参考第一对象数据执行第一数据处理指令,得到目标第一对象数据,以提高数据处理效率。

[0169] 其中,该一个或多个程序被存储在上述存储器中,且被配置由上述处理器执行,该一个或多个程序包括用于执行上述方法实施例中任一步骤的指令。

[0170] 其中,处理器例如可以是中央处理器(Central Processing Unit,CPU),通用处理器,数字信号处理器(Digital Signal Processor,DSP),专用集成电路(Application-Specific Integrated Circuit,ASIC),现场可编程门阵列(Field Programmable Gate Array,FPGA)或者其他可编程逻辑器件、晶体管逻辑器件、硬件部件或者其任意组合。其可以实现或执行结合本申请公开内容所描述的各种示例性的逻辑方框,单元和电路。处理器也可以是实现计算功能的组合,例如包含一个或多个微处理器组合,DSP和微处理器的组合等等。通信单元可以是通信接口、收发器、收发电路等,存储单元可以是存储器。

[0171] 存储器可以是易失性存储器或非易失性存储器,或可包括易失性和非易失性存储器两者。其中,非易失性存储器可以是只读存储器(read-only memory,ROM)、可编程只读存储器(programmable ROM,PROM)、可擦除可编程只读存储器(erasable PROM,EPROM)、电可擦除可编程只读存储器(electrically EPROM,EEPROM)或闪存。易失性存储器可以是随机存取存储器(random access memory,RAM),其用作外部高速缓存。通过示例性但不是限制性说明,许多形式的随机存取存储器(random access memory,RAM)可用,例如静态随机存取存储器(static RAM,SRAM)、动态随机存取存储器(DRAM)、同步动态随机存取存储器(synchronous DRAM,SDRAM)、双倍数据速率同步动态随机存取存储器(doubledata rate SDRAM,DDR SDRAM)、增强型同步动态随机存取存储器(enhanced SDRAM,ESDRAM)、同步连接动态随机存取存储器(synchlink DRAM,SLDRAM)和直接内存总线随机存取存储器(direct rambus RAM,DR RAM)。

[0172] 可以理解的是,电子设备可包括比上述结构框图中更多或更少的结构元件,例如,包括电源模块、物理按键、Wi-Fi模块、扬声器、蓝牙模块、传感器、显示模块等,在此不进行限定。可以理解,该电子设备可以是目标设备。

[0173] 上述主要从方法侧执行过程的角度对本申请实施例的方案进行了介绍。可以理解的是,电子设备为了实现上述功能,其包含了执行各个功能相应的硬件结构和/或软件模块。本领域技术人员应该很容易意识到,结合本文中所提供的实施例描述的各示例的单元及算法步骤,本申请能够以硬件或硬件和计算机软件的结合形式来实现。某个功能究竟以硬件还是计算机软件驱动硬件的方式来执行,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

现不应认为超出本申请的范围。

[0174] 本申请实施例可以根据上述方法示例对电子设备进行功能单元的划分,例如,可以对应各个功能划分各个功能单元,也可以将两个或两个以上的功能集成在一个处理单元中。上述集成的单元既可以采用硬件的形式实现,也可以采用软件功能单元的形式实现。需要说明的是,本申请实施例中对单元的划分是示意性的,仅仅为一种逻辑功能划分,实际实现时可以有另外的划分方式。

[0175] 在采用对应各个功能划分各个功能模块的情况下,图10为本申请实施例提供的一种针对内存受限的数据处理装置的功能模块组成框图,应用于数据解析转换机,目标设备包括所述数据解析转换机、目标应用程序、目标内存和目标磁盘,所述针对内存受限的数据处理装置10包括获取模块101、确定模块102、填充模块103、创建模块104以及执行模块105,其中:

[0176] 所述获取模块101,用于获取所述目标应用程序对应的第一对象;所述第一对象存储在所述目标内存;所述第一对象包括第一对象数据和第一对象定义;

[0177] 所述确定模块102,用于确定所述第一对象定义对应的第一引用地址,并根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址;所述第二引用地址用于指向所述第一对象定义;

[0178] 所述填充模块103,用于根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据,以节省所述目标内存的数据空间;

[0179] 所述创建模块104,用于在所述目标磁盘中创建自恢复日志文件,并根据所述自恢复日志文件确定自恢复日志文件写操作机制;所述自恢复日志文件写操作机制用于存储数据处理操作,以确保数据在系统崩溃或故障后能够恢复;

[0180] 所述获取模块101,还用于获取所述参考第一对象数据关联的第一数据处理指令;

[0181] 所述执行模块105,用于根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令,得到目标第一对象数据,以提高数据处理效率。

[0182] 可选的,在所述根据预设的编码函数对所述第一引用地址进行压缩,得到第二引用地址方面,所述确定模块102还具体用于:

[0183] 获取所述第一引用地址对应的内存区域的起始偏移量;

[0184] 将所述第一引用地址与所述起始偏移量进行相减,得到参考引用地址;

[0185] 将所述参考引用地址无符号右移a位以进行地址压缩,得到所述第二引用地址;a为正整数。

[0186] 可选的,所述第一对象数据包括多个子数据;所述预设存储结构包括多层数据空间,每一层数据空间的内存大小相同;在所述根据预设存储结构将所述第一对象数据按照预设存储规则进行填充,得到参考第一对象数据方面,所述填充模块103还具体用于:

[0187] 若所述第一对象数据无继承层次,则将所述多个子数据按照预设的第一排序规则进行排序,得到第一子数据序列;

[0188] 将所述第一子数据序列按照先后顺序填充至所述多层数据空间,得到所述参考第一对象数据;

[0189] 若所述第一对象数据包括多个继承层次,则将所述多个继承层次中每一继承层次的子数据按照所述第一排序规则进行排序,得到多个参考第二子数据序列;每一继承层次

对应一个参考第二子数据序列；

[0190] 根据继承层次从上到下的顺序对所述多个参考第二子数据序列进行排序，得到目标第二子数据序列；

[0191] 将所述目标第二子数据序列按照先后顺序填充至所述多层数据空间，得到所述参考第一对象数据。

[0192] 可选的，所述填充模块103还具体用于：

[0193] 若所述第一对象数据包括多个继承层次，则获取继承层次i的至少一个子数据；所述继承层次i为所述多个继承层次中任意一个继承层次；

[0194] 若所述至少一个子数据对应的数据类型中存在预设第一类型，且继承层次i-1对应的数据空间未满，则将所述至少一个子数据按照预设的第二排序规则进行排序，得到参考第三子数据序列；

[0195] 将所述参考第三子数据序列按照先后顺序填充至所述继承层次i-1对应的数据空间；

[0196] 当所述继承层次i-1对应的数据空间无法存入数据时，将未存入数据空间的所有子数据按照所述第一排序规则进行排序，得到参考第四子数据序列；

[0197] 将所述参考第四子数据序列按照先后顺序填充至所述多层数据空间，得到所述参考第一对象数据。

[0198] 可选的，所述自恢复日志文件包括写入指针和检查指针，所述写入指针用于指示逆时针写入数据的起始位置，所述检查指针用于指示所述自恢复日志文件中数据的起始位置，在所述目标磁盘中创建自恢复日志文件，并根据所述自恢复日志文件确定自恢复日志文件写操作机制方面，所述创建模块104还具体用于：

[0199] 在所述目标磁盘中获取k个日志文件；所述k个日志文件中每一日志文件大小一致；k为正整数；

[0200] 将所述k个日志文件按照预设的环形结构进行整合，得到所述自恢复日志文件；

[0201] 当所述自恢复日志文件执行参考磁盘顺序写操作时，对所述写入指针和所述检查指针的位置进行检查；

[0202] 当所述检查指针在所述写入指针之后，则确定所述自恢复日志文件中数据未满，继续执行所述参考磁盘顺序写操作；

[0203] 当所述检查指针与所述写入指针重合，则确定所述自恢复日志文件中数据已满，获取所述检查指针的第一位置；

[0204] 根据预设移动距离控制所述检查指针逆时针向前移动，得到所述检查指针的第二位置；

[0205] 确定所述第一位置和所述第二位置之间的数据为待处理数据，并对所述待处理数据进行处理，得到空余的参考日志文件空间；

[0206] 根据所述参考日志文件空间继续执行所述参考磁盘顺序写操作。

[0207] 可选的，所述自恢复日志文件包括第一自恢复日志文件，在所述根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令，得到目标第一对象数据方面，所述执行模块105还具体用于：

[0208] 根据所述参考第一对象数据确定第一原始数据页；

[0209] 根据所述第一数据处理指令对所述第一原始数据页中的所述参考第一对象数据进行修改,得到第一数据变更页;所述第一数据变更页包括变更第一对象数据;

[0210] 通过对所述第一自恢复日志文件执行第一磁盘顺序写操作,将所述第一数据处理指令存储至所述第一自恢复日志文件;

[0211] 当针对所述第一自恢复日志文件的读写操作满足预设第一条件时,通过对所述第一数据变更页执行第一磁盘随机写操作,将所述变更第一对象数据写入所述目标磁盘,得到所述目标第一对象数据。

[0212] 可选的,所述自恢复日志文件还包括第二自恢复日志文件,所述执行模块105还具体用于:

[0213] 获取所述目标应用程序对应的第二对象;所述第二对象存储在所述目标磁盘;

[0214] 确定所述第二对象对应的第二对象数据关联的第二数据处理指令;所述第二数据处理指令包括至少一个数据操作;

[0215] 在所述目标内存中创建数据处理缓存页;

[0216] 将所述第二数据处理指令存储至所述数据处理缓存页,并对所述第二数据处理指令中至少一个数据操作进行整合,得到目标数据处理指令;

[0217] 通过对所述第二自恢复日志文件执行第二磁盘顺序写操作,将所述目标数据处理指令存储至所述第二自恢复日志文件。

[0218] 可选的,在所述将所述目标数据处理指令存储至所述第二自恢复日志文件之后,所述执行模块105还具体用于:

[0219] 通过对所述目标磁盘执行第二磁盘随机写操作,将所述第二对象数据从所述目标磁盘加载到所述目标内存,得到第二原始数据页;

[0220] 根据所述目标数据处理指令对所述第二原始数据页中的所述第二对象数据进行处理,得到第二数据变更页;所述第二数据变更页包括变更第二对象数据;

[0221] 通过对所述第二自恢复日志文件执行第三磁盘顺序写操作,将所述目标数据处理指令存储至所述第二自恢复日志文件;

[0222] 当针对所述第二自恢复日志文件的读写操作满足预设第二条件时,通过对所述第二数据变更页执行第三磁盘随机写操作,将所述变更第二对象数据写入所述目标磁盘,得到目标第二对象数据。

[0223] 可见,通过对内存数据的存储进行优化,并对内存计算和文件交互机制进行改进,以提高内存利用率并提升大量数据计算的处理性能。

[0224] 需要说明的是,各个操作的具体实现可以采用上述所示的方法实施例的相应描述,针对内存受限的数据处理装置10可以用于执行本申请上述方法实施例,对此不再赘述。

[0225] 本申请实施例还提供了一种计算机可读存储介质,其中,该计算机可读存储介质存储用于电子数据交换的计算机程序,该计算机程序使得计算机执行如上述方法实施例中记载的任一方法的部分或全部步骤,上述计算机包括电子设备。

[0226] 本申请实施例还提供了一种计算机程序产品,上述计算机程序产品包括存储了计算机程序的非瞬时性计算机可读存储介质,上述计算机程序可操作来使计算机执行如上述方法实施例中记载的任一方法的部分或全部步骤。该计算机程序产品可以为一个软件安装包,上述计算机包括电子设备。

[0227] 需要说明的是,对于上述的各个实施例,为了简单描述,将其都表述为一系列的动作组合。本领域技术人员应该知悉,本申请不受所描述的动作顺序的限制,因为本申请实施例中的某些步骤可以采用其他顺序或者同时进行。另外,本领域技术人员也应该知悉,说明书中所描述的实施例均属于优选实施例,所涉及的动作、步骤、模块或单元等并不一定是本申请实施例所必须的。

[0228] 在上述实施例中,本申请实施例对各个实施例的描述都各有侧重,某个实施例中没有详述的部分,可以参见其他实施例的相关描述。

[0229] 本领域普通技术人员可以理解实现上述实施例方法中的全部或部分流程,该流程可以由计算机程序来指令相关的硬件完成,该程序可存储于计算机可读取存储介质中,该程序在执行时,可包括如上述各方法实施例的流程。而前述的存储介质包括:ROM或随机存储记忆体RAM、磁碟或者光盘等各种可存储程序代码的介质。

[0230] 本申请实施例所描述的方法或者算法的步骤可以以硬件的方式来实现,也可以是由处理器执行软件指令的方式来实现。软件指令可以由相应的软件模块组成,软件模块可以被存放于RAM、闪存、ROM、EPROM、电可擦可编程只读存储器(electrically EPROM,EEPROM)、寄存器、硬盘、移动硬盘、只读光盘(CD-ROM)或者本领域熟知的任何其它形式的存储介质中。一种示例性的存储介质耦合至处理器,从而使处理器能够从该存储介质读取信息,且可向该存储介质写入信息。当然,存储介质也可以是处理器的组成部分。处理器和存储介质可以位于ASIC中。另外,该ASIC可以位于终端设备或管理设备中。当然,处理器和存储介质也可以作为分立组件存在于终端设备或管理设备中。

[0231] 本领域技术人员应该可以意识到,在上述一个或多个示例中,本申请实施例所描述的功能可以全部或部分地通过软件、硬件、固件或者其任意组合来实现。当使用软件实现时,可以全部或部分地以计算机程序产品的形式实现。该计算机程序产品包括一个或多个计算机指令。在计算机上加载和执行该计算机程序指令时,全部或部分地产生按照本申请实施例所述的流程或功能。该计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。该计算机指令可以存储在计算机可读存储介质中,或者从一个计算机可读存储介质向另一个计算机可读存储介质传输。例如,该计算机指令可以从一个网站站点、计算机、服务器或数据中心通过有线(例如同轴电缆、光纤、数字用户线(digital subscriber line,DSL))或无线(例如红外、无线、微波等)方式向另一个网站站点、计算机、服务器或数据中心进行传输。该计算机可读存储介质可以是计算机能够存取的任何可用介质或者是包含一个或多个可用介质集成的服务器、数据中心等数据存储设备。该可用介质可以是磁性介质(例如,软盘、硬盘、磁带)、光介质(例如,数字视频光盘(digital video disc,DVD))、或者半导体介质(例如,固态硬盘(solid state disk,SSD))等。

[0232] 上述实施例中描述的各个装置、产品包含的各个模块/单元,其可以是软件模块/单元,也可以是硬件模块/单元,或者也可以部分是软件模块/单元,部分是硬件模块/单元。例如,对于应用于或集成于芯片的各个装置、产品,其包含的各个模块/单元可以都采用电路等硬件的方式实现,或者,至少部分模块/单元可以采用软件程序的方式实现,该软件程序运行于芯片内部集成的处理器,剩余的(如果有)部分模块/单元可以采用电路等硬件方式实现;对于应用于或集成于芯片模组的各个装置、产品,其包含的各个模块/单元可以都采用电路等硬件的方式实现,不同的模块/单元可以位于芯片模组的同一组件(例如芯片、

电路模块等)或者不同组件中,或者,至少部分模块/单元可以采用软件程序的方式实现,该软件程序运行于芯片模组内部集成的处理器,剩余的(如果有)部分模块/单元可以采用电路等硬件方式实现;对于应用于或集成于终端设备的各个装置、产品,其包含的各个模块/单元可以都采用电路等硬件的方式实现,不同的模块/单元可以位于终端设备内同一组件(例如,芯片、电路模块等)或者不同组件中,或者,至少部分模块/单元可以采用软件程序的方式实现,该软件程序运行于终端设备内部集成的处理器,剩余的(如果有)部分模块/单元可以采用电路等硬件方式实现。

[0233] 以上所述的具体实施方式,对本申请实施例的目的、技术方案和有益效果进行了进一步详细说明,所应理解的是,以上所述仅为本申请实施例的具体实施方式而已,并不用于限定本申请实施例的保护范围,凡在本申请实施例的技术方案的基础之上,所做的任何修改、等同替换、改进等,均应包括在本申请实施例的保护范围之内。

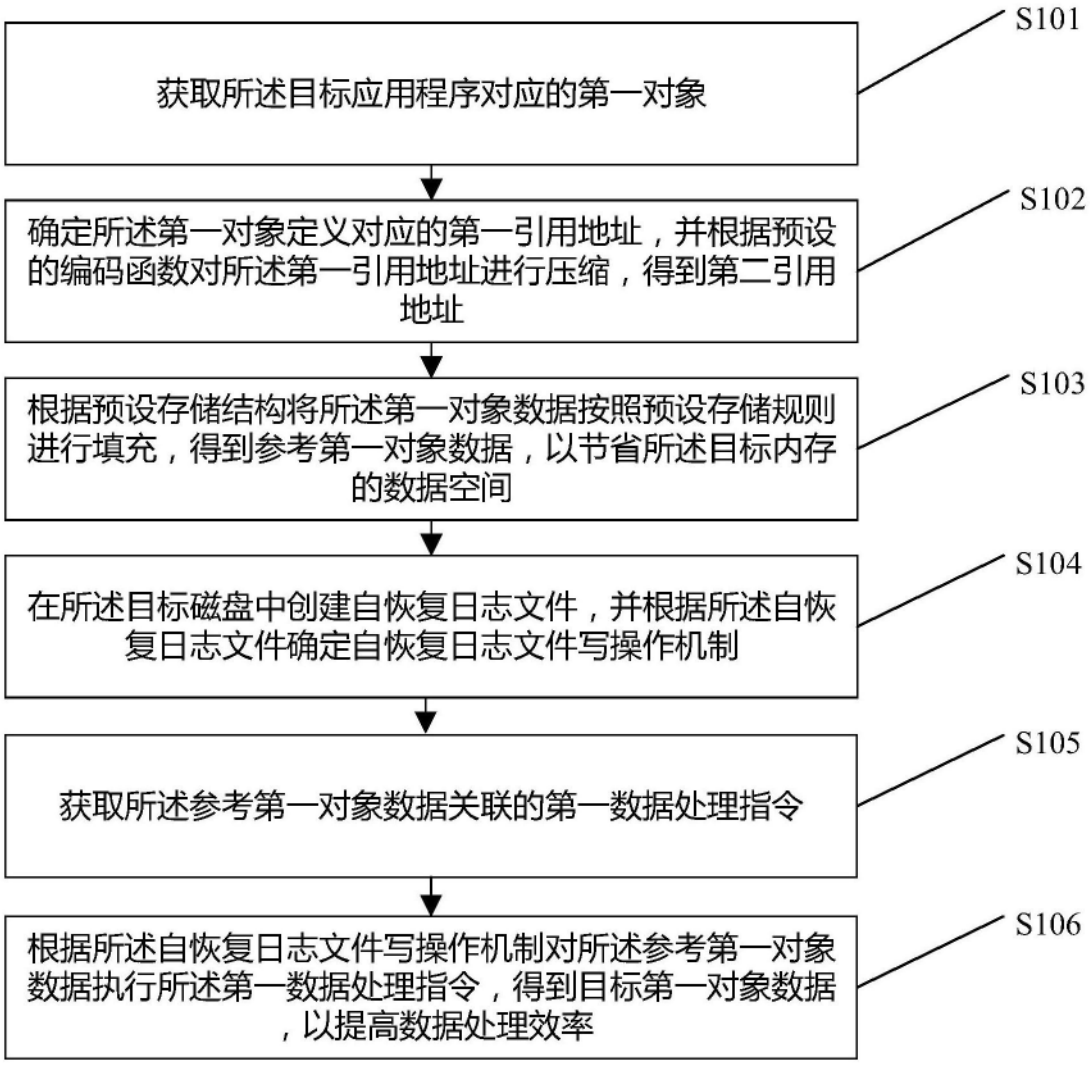

S101
获取所述目标应用程序对应的第一对象
S102
确定所述第一对象定义对应的第一引用地址，并根据预设的编码函数对所述第一引用地址进行压缩，得到第二引用地址
S103
根据预设存储结构将所述第一对象数据按照预设存储规则进行填充，得到参考第一对象数据，以节省所述目标内存的数据空间
S104
在所述目标磁盘中创建自恢复日志文件，并根据所述自恢复日志文件确定自恢复日志文件写操作机制
S105
获取所述参考第一对象数据关联的第一数据处理指令
S106
根据所述自恢复日志文件写操作机制对所述参考第一对象数据执行所述第一数据处理指令，得到目标第一对象数据，以提高数据处理效率

图1

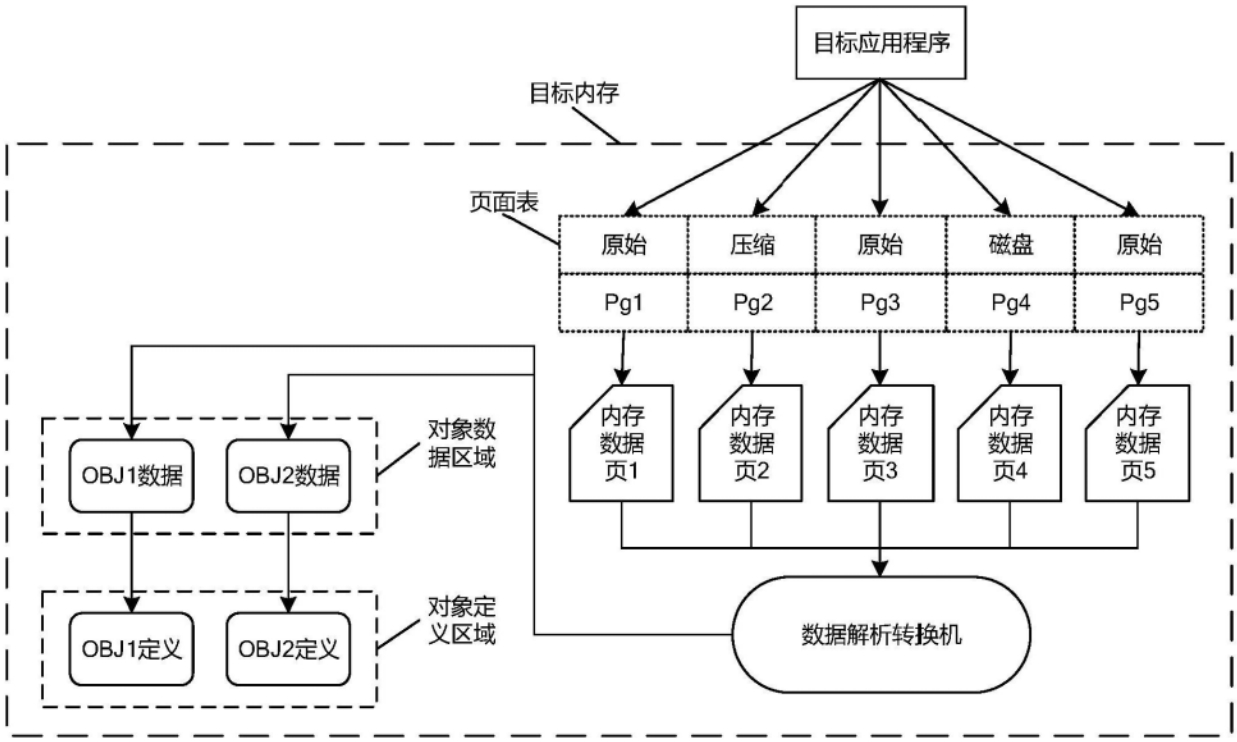

目标应用程序
目标内存
页面表
原始
压缩
原始
磁盘
原始
Pg1
Pg2
Pg3
Pg4
Pg5
内存数据页1
内存数据页2
内存数据页3
内存数据页4
内存数据页5
对象数据区域
OBJ1数据
OBJ2数据
对象定义区域
OBJ1定义
OBJ2定义
数据解析转换机

图2

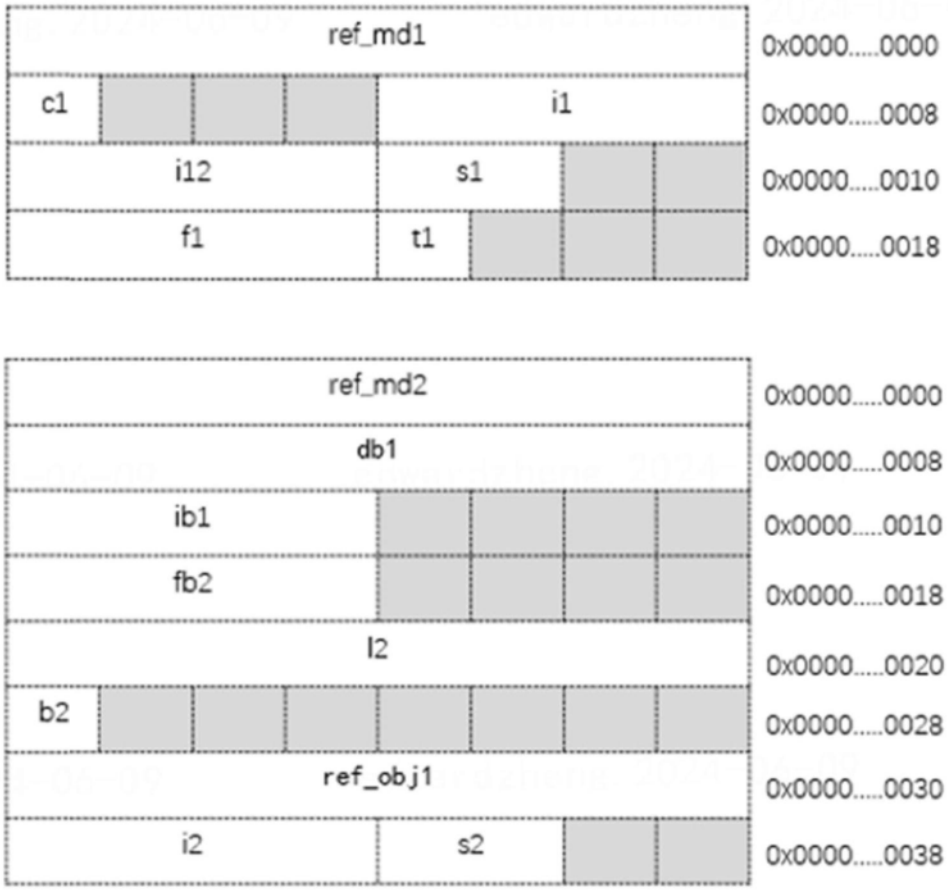

ref_md1
c1
i1
i12
s1
f1
t1
0x0000.....0000
0x0000.....0008
0x0000.....0010
0x0000.....0018
ref_md2
db1
ib1
fb2
l2
b2
ref_obj1
i2
s2
0x0000.....0000
0x0000.....0008
0x0000.....0010
0x0000.....0018
0x0000.....0020
0x0000.....0028
0x0000.....0030
0x0000.....0038

图3

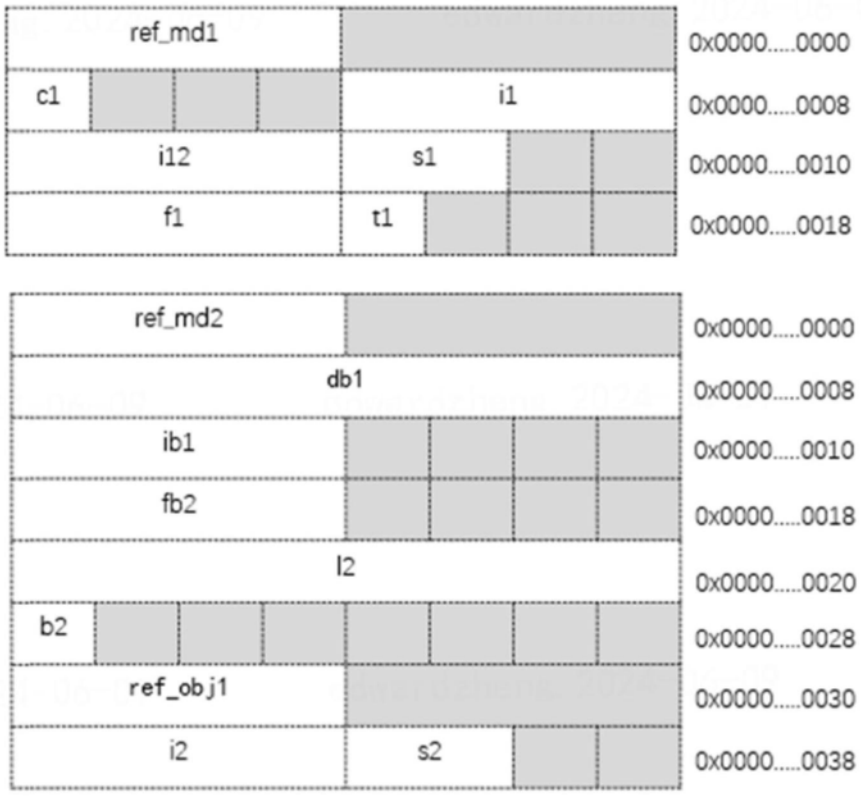

ref_md1
0x0000.....0000
c1
i1
0x0000.....0008
i12
s1
0x0000.....0010
f1
t1
0x0000.....0018
ref_md2
0x0000.....0000
db1
0x0000.....0008
ib1
0x0000.....0010
fb2
0x0000.....0018
l2
0x0000.....0020
b2
0x0000.....0028
ref_obj1
0x0000.....0030
i2
s2
0x0000.....0038

图4

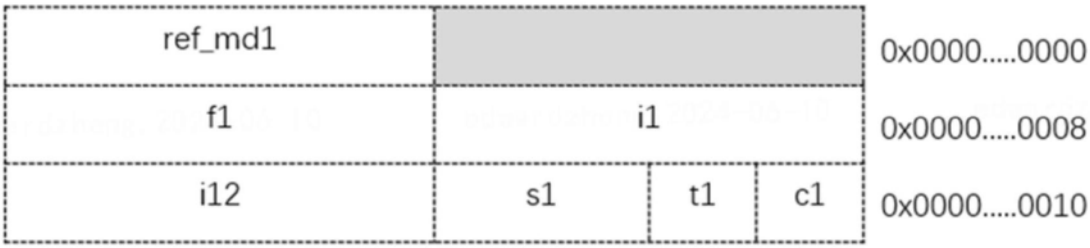

ref_md1
0x0000.....0000
f1
i1
0x0000.....0008
i12
s1
t1
c1
0x0000.....0010

图5

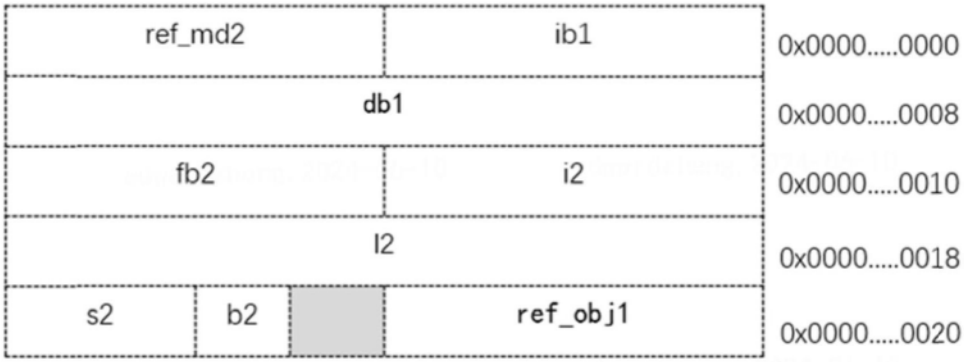

ref_md2
ib1
0x0000.....0000
db1
0x0000.....0008
fb2
i2
0x0000.....0010
l2
0x0000.....0018
s2
b2
ref_obj1
0x0000.....0020

图6

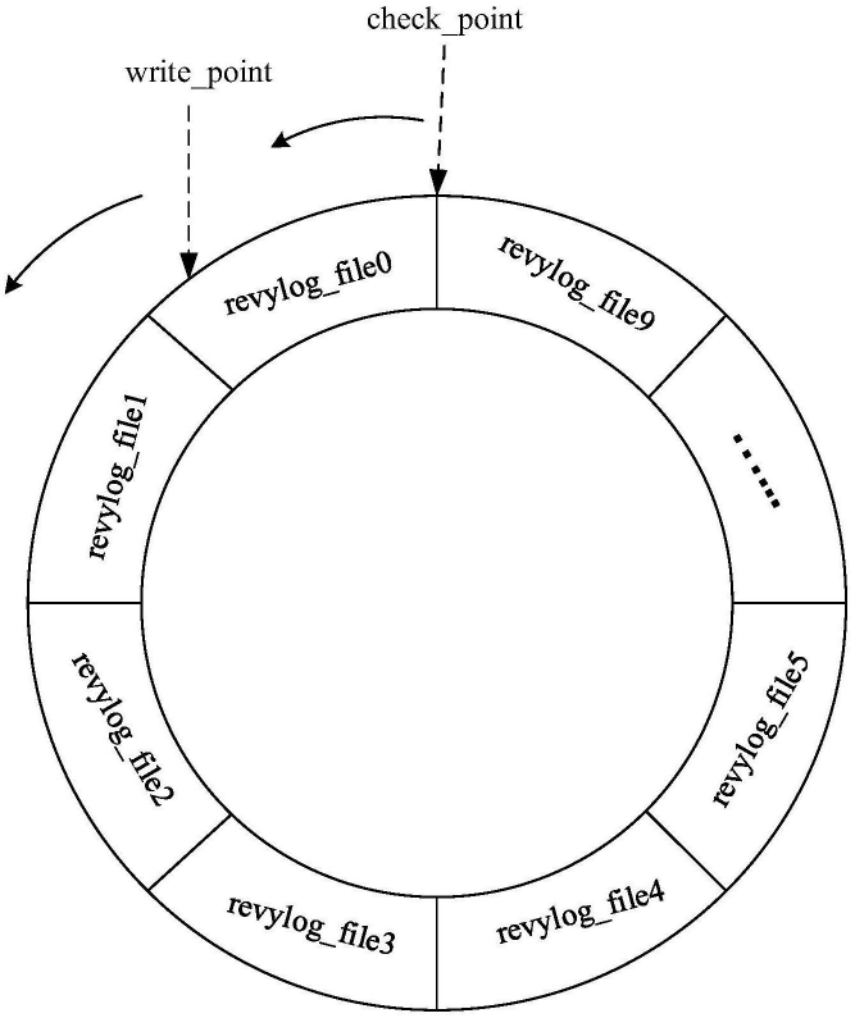

check_point
write_point
revylog_file0
revylog_file9
revylog_file1
.....
revylog_file2
revylog_file5
revylog_file3
revylog_file4

图7

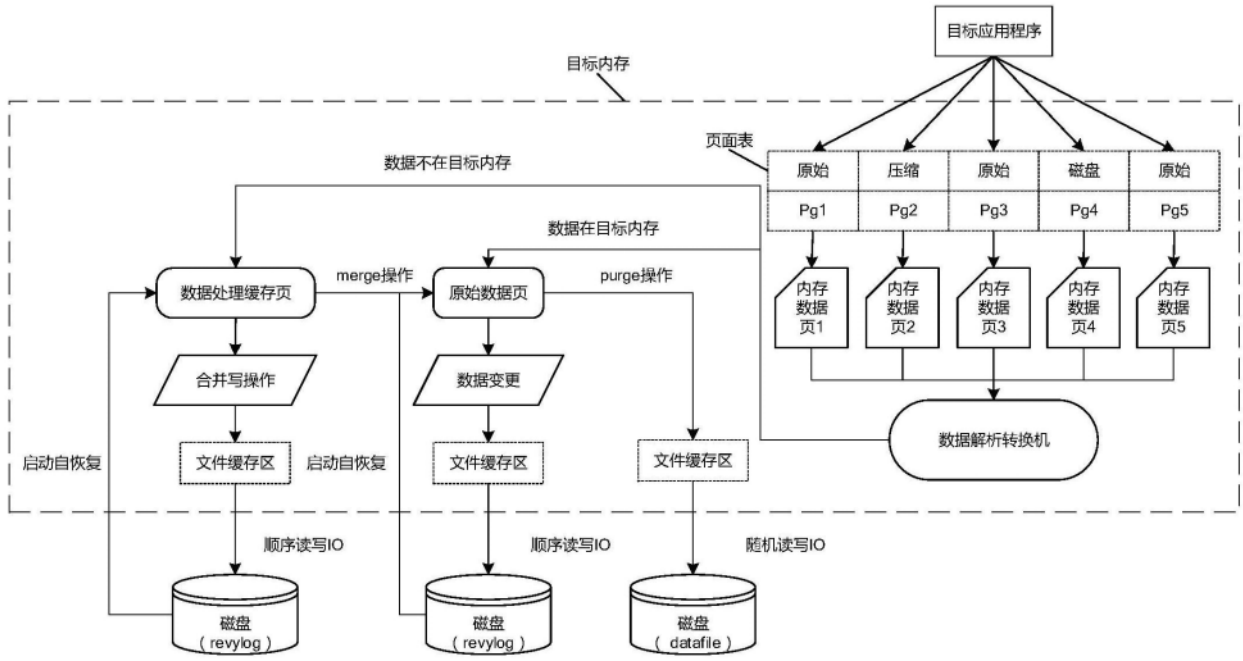

目标应用程序
目标内存
页面表
原始
压缩
原始
磁盘
原始
Pg1
Pg2
Pg3
Pg4
Pg5
数据不在目标内存
数据在目标内存
数据处理缓存页
merge操作
原始数据页
purge操作
内存数据页1
内存数据页2
内存数据页3
内存数据页4
内存数据页5
合并写操作
数据变更
数据解析转换机
启动自恢复
文件缓存区
启动自恢复
文件缓存区
文件缓存区
顺序读写IO
顺序读写IO
随机读写IO
磁盘（revylog）
磁盘（revylog）
磁盘（datafile）

图8

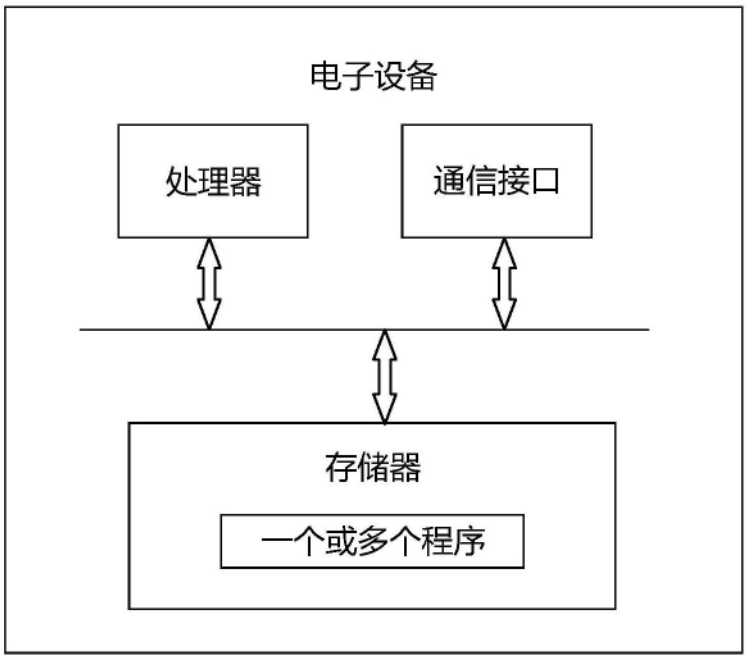

电子设备
处理器
通信接口
存储器
一个或多个程序

图9

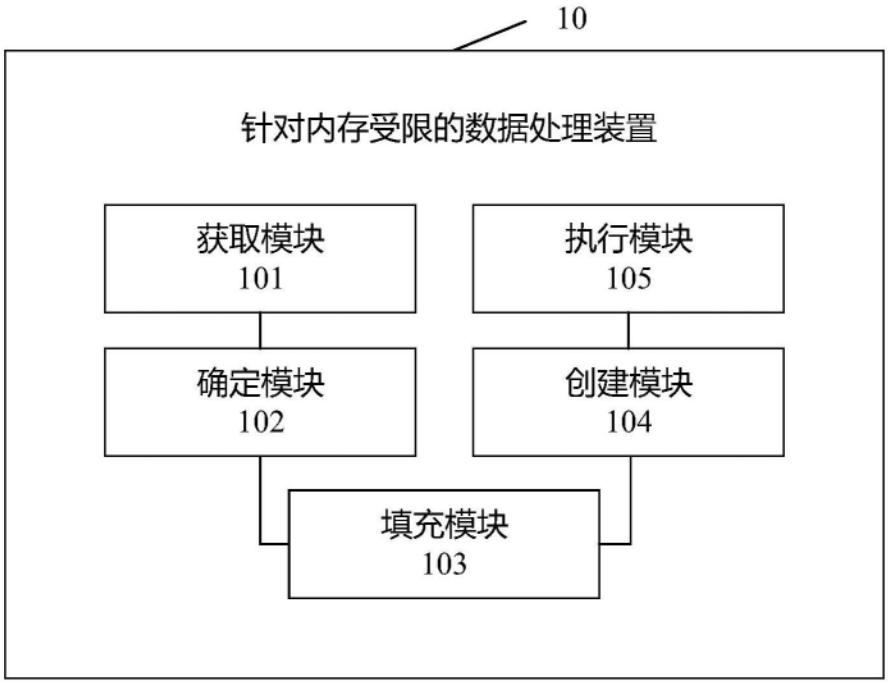

10
针对内存受限的数据处理装置
获取模块
101
执行模块
105
确定模块
102
创建模块
104
填充模块
103

图10