



(12) 发明专利

(10) 授权公告号 CN 117667853 B

(45) 授权公告日 2024. 05. 03

(21) 申请号 202410129220.8

G06F 16/14 (2019.01)

(22) 申请日 2024.01.30

(56) 对比文件

(65) 同一申请的已公布的文献号

CN 114968942 A, 2022.08.30

申请公布号 CN 117667853 A

US 2014/0025712 A1, 2014.01.23

(43) 申请公布日 2024.03.08

CN 116185308 A, 2023.05.30

(73) 专利权人 苏州元脑智能科技有限公司

CN 109597903 A, 2019.04.09

地址 215000 江苏省苏州市吴中经济开发

US 2020/0245040 A1, 2020.07.30

区郭巷街道官浦路1号9幢

审查员 马婷婷

(72) 发明人 王继玉 陈培 荆荣讯

(74) 专利代理机构 北京三聚阳光知识产权代理

有限公司 11250

专利代理师 李博洋

(51) Int. Cl.

G06F 16/13 (2019.01)

G06F 16/16 (2019.01)

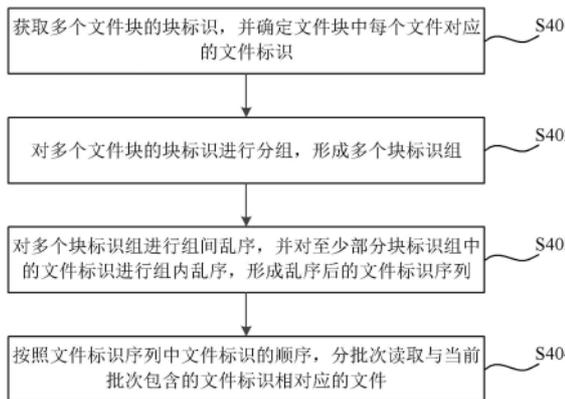
权利要求书4页 说明书25页 附图15页

(54) 发明名称

数据读取方法、装置、计算机设备及存储介质

(57) 摘要

本发明涉及数据处理技术领域,公开了数据读取方法、装置、计算机设备及存储介质,该方法包括:获取多个文件块的块标识,并确定所述文件块中每个文件对应的文件标识;所述文件块是对文件数据集中的文件进行聚合所生成的数据块;对所述多个文件块的块标识进行分组,形成多个块标识组;对所述多个块标识组进行组间乱序,并对至少部分所述块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列;按照所述文件标识序列中文件标识的顺序,分批次读取与当前批次包含的文件标识相对应的文件。本发明可以方便快速地实现对整个文件数据集的乱序,在保证模型训练准确度的同时,可以提高乱序效率,能够保证读取效率。



1. 一种数据读取方法,其特征在于,所述方法包括:

获取多个文件块的块标识,并确定所述文件块中每个文件对应的文件标识;所述文件块是对文件数据集中的文件进行聚合所生成的数据块;

对所述多个文件块的块标识进行分组,形成多个块标识组;

对所述多个块标识组进行组间乱序,并对至少部分所述块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列;

按照所述文件标识序列中文件标识的顺序,分批次读取与当前批次包含的文件标识相对应的文件;

其中,所述文件标识为相对应的文件在所述多个文件块所包含的所有文件中的索引;

所述对所述多个块标识组进行组间乱序,并对至少部分所述块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列,包括:

构建二维数组;所述二维数组中的第一维元素表示相应的块标识组,所述第一维元素对应的第二维元素表示相应块标识组中的文件标识;

对所述二维数组中的多个所述第一维元素进行乱序,并对至少部分所述第一维元素对应的第二维元素进行乱序;

按照所述二维数组中元素乱序后的顺序,将乱序后的所述二维数组转换为一维的全局索引列表;所述全局索引列表作为文件标识序列。

2. 根据权利要求1所述的方法,其特征在于,分批次读取的文件用于模型训练,且所述方法还包括:

在模型训练的第一次迭代过程中,以所述块标识组为单位生成包含多个文件的描述数据的文件列表;

对所述文件列表中的描述数据进行乱序,并将乱序后的文件列表按照所述块标识组的顺序,形成包括多个文件列表中描述数据的全局文件列表;

将文件的描述数据在所述全局文件列表中的位置下标作为相应文件的索引。

3. 根据权利要求1所述的方法,其特征在于,所述对所述多个文件块的块标识进行分组,形成多个块标识组,包括:

根据所述文件块中的文件数量,确定分组大小;所述文件块中的文件数量与所述分组大小之间为负相关关系;

按照所述分组大小对所述多个文件块的块标识进行分组,形成多个块标识组。

4. 根据权利要求3所述的方法,其特征在于,

所述文件块中的文件数量为在文件块大小的限制下所聚合的文件数量,所述分组大小为根据所述文件数据集的规模所设置的大小;

或者,所述文件块中的文件数量为 2^m ,所述分组大小为 2^n ;m、n均为整数。

5. 根据权利要求1所述的方法,其特征在于,所述读取与当前批次包含的文件标识相对应的文件,包括:

对当前批次包含的文件标识相对应的文件进行寻址,确定相应的寻址信息;

根据所述寻址信息从命中的文件块中读取相应的文件。

6. 根据权利要求5所述的方法,其特征在于,所述对当前批次包含的文件标识相对应的文件进行寻址,确定相应的寻址信息,包括:

确定当前批次包含的文件标识命中的目标文件,并生成所述目标文件的寻址信息对象;所述寻址信息对象包括所述目标文件在相应文件块中的位置信息。

7.根据权利要求6所述的方法,其特征在于,所述生成所述目标文件的寻址信息对象,包括:

以批量寻址的方式,批量生成多个所述目标文件的寻址信息对象;

或者,根据构建的全局字典进行寻址,确定多个所述目标文件的寻址信息对象;所述全局字典用于维护所述目标文件所在的文件块的所有文件的寻址信息对象。

8.根据权利要求7所述的方法,其特征在于,所述以批量寻址的方式,批量生成多个所述目标文件的寻址信息对象,包括:

确定当前批次包含的文件标识命中的目标文件块,创建所述目标文件块的块对象;

遍历所述目标文件块中的文件的头信息,并判断所述文件的头信息与所述当前批次包含的文件标识是否相匹配;

在所述文件的头信息与所述当前批次包含的文件标识相匹配时,根据所述文件的头信息确定所述文件在所述目标文件块中的位置信息;所述位置信息包括文件的句柄、在所述目标文件块中的位置和偏移量;

根据所述位置信息生成相匹配文件的寻址信息对象,并将所述寻址信息对象返回至块对象;

在所述块对象中的寻址信息对象的数量与所述当前批次包含的文件标识中属于所述目标文件块的文件标识数量一致时,停止遍历所述目标文件块。

9.根据权利要求7所述的方法,其特征在于,所述根据构建的全局字典进行寻址,确定多个所述目标文件的寻址信息对象,包括:

判断当前批次包含的文件标识命中的目标文件块是否为第一次命中;

在第一次命中所述目标文件块的情况下,创建所述目标文件块的块对象;

遍历所述目标文件块中所有文件的头信息,并根据所述文件的头信息确定所述文件在所述目标文件块中的位置信息;所述位置信息包括文件的句柄、在所述目标文件块中的位置和偏移量;

根据所述位置信息确定所有文件的寻址信息对象,并将所有文件的所述寻址信息对象返回至块对象;

将所述块对象放入至全局字典,并从所述全局字典中读取当前批次包含的文件标识所对应的目标文件的寻址信息对象;

在非第一次命中所述目标文件块的情况下,从所述全局字典中读取当前批次包含的文件标识所对应的目标文件的寻址信息对象。

10.根据权利要求9所述的方法,其特征在于,还包括:

在所述全局字典中的寻址信息对象对应的目标文件被读取之后,删除被读取的目标文件所对应的寻址信息对象;

在所述块对象中的任意寻址信息对象对应的目标文件均被读取之后,删除相应的块对象。

11.根据权利要求7所述的方法,其特征在于,所述生成所述目标文件的寻址信息对象,包括:

确定当前批次包含的文件标识命中的目标文件块中文件的第一数量,并确定当前批次包含的文件标识命中的、属于所述目标文件块的目标文件的第二数量;

在所述第一数量超过第一阈值、且所述第二数量超过第二阈值的情况下,以批量寻址的方式,批量生成多个所述目标文件的寻址信息对象;

在所述第一数量未超过第一阈值、且所述第二数量未超过第二阈值的情况下,根据构建的全局字典进行寻址,确定多个所述目标文件的寻址信息对象。

12. 根据权利要求5所述的方法,其特征在于,所述对当前批次包含的文件标识相对应的文件进行寻址,确定相应的寻址信息,包括:

根据当前批次包含的文件标识确定当前批次命中的目标文件所属的文件块;

将属于同一文件块的目标文件归为一组,生成包含至少一个目标文件所对应文件标识的目标文件列表;

以所述目标文件列表为单位,分别确定每个所述目标文件列表所对应的目标文件的寻址信息。

13. 根据权利要求12所述的方法,其特征在于,还包括:

开启多个工作进程,基于所述工作进程读取相应批次的文件;不同的所述工作进程用于读取不同批次的文件,且在所述工作进程读取完当前批次的文件后,所述工作进程用于读取下一批次的文件,直至读取完所述文件数据集;

在当前批次的文件读取过程中,开启与命中的文件块数量相匹配的工作线程,每个所述工作线程分别用于确定相应文件块中文件的寻址信息,并读取相应的文件。

14. 根据权利要求5所述的方法,其特征在于,所述根据所述寻址信息从命中的文件块中读取相应的文件,包括:

判断当前批次包含的文件标识命中的目标文件块是否为第一次命中;

在第一次命中所述目标文件块的情况下,对所述目标文件块进行缓存,并根据所述寻址信息从缓存的目标文件块中读取相应的目标文件;

在非第一次命中所述目标文件块的情况下,根据所述寻址信息从缓存的目标文件块中读取相应的目标文件。

15. 根据权利要求14所述的方法,其特征在于,所述根据所述寻址信息从缓存的目标文件块中读取相应的目标文件,包括:

在所述目标文件为图片数据的情况下,将所述目标文件的寻址信息对象传给图像读取函数,以生成图片文件对象;根据所述图片文件对象读取相应的图片数据,并进行格式转换;

在所述目标文件为文本数据的情况下,根据所述目标文件的寻址信息对象读取相应的文本数据,并进行格式转换。

16. 一种数据读取装置,其特征在于,所述装置包括:

获取模块,用于获取多个文件块的块标识,并确定所述文件块中每个文件对应的文件标识;所述文件块是对文件数据集中的文件进行聚合所生成的数据块;

分组模块,用于对所述多个文件块的块标识进行分组,形成多个块标识组;

乱序模块,用于对所述多个块标识组进行组间乱序,并对至少部分所述块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列;

读取模块,用于按照所述文件标识序列中文件标识的顺序,分批次读取与当前批次包含的文件标识相对应的文件;

其中,所述文件标识为相对应的文件在所述多个文件块所包含的所有文件中的索引;

所述乱序模块对所述多个块标识组进行组间乱序,并对至少部分所述块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列,包括:

构建二维数组;所述二维数组中的第一维元素表示相应的块标识组,所述第一维元素对应的第二维元素表示相应块标识组中的文件标识;

对所述二维数组中的多个所述第一维元素进行乱序,并对至少部分所述第一维元素对应的第二维元素进行乱序;

按照所述二维数组中元素乱序后的顺序,将乱序后的所述二维数组转换为一维的全局索引列表;所述全局索引列表作为文件标识序列。

17.一种计算机设备,其特征在于,包括:

存储器和处理器,所述存储器和所述处理器之间互相通信连接,所述存储器中存储有计算机指令,所述处理器通过执行所述计算机指令,从而执行权利要求1至15中任一项所述的数据读取方法。

18.一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上存储有计算机指令,所述计算机指令用于使计算机执行权利要求1至15中任一项所述的数据读取方法。

19.一种计算机程序产品,包括计算机程序/指令,其特征在于,所述计算机程序/指令被处理器执行时实现权利要求1至15中任一项所述的数据读取方法。

数据读取方法、装置、计算机设备及存储介质

技术领域

[0001] 本发明涉及数据处理技术领域,具体涉及数据读取方法、装置、计算机设备及存储介质。

背景技术

[0002] AI(Artificial Intelligence,人工智能)模型训练,初期阶段是将数据集本地化后,再进行训练,这一般需要花费很长的时间。而随着大数据和AI技术的发展,复杂场景的AI模型训练,一般需要海量小文件作为训练集。

[0003] 目前,海量小文件本地化后,其读取方式效率较低,进而会影响训练效率。

发明内容

[0004] 有鉴于此,本发明提供了一种数据读取方法、装置、计算机设备及存储介质,以解决读取效率较低的问题。

[0005] 第一方面,本发明提供了一种数据读取方法,包括:

[0006] 获取多个文件块的块标识,并确定所述文件块中每个文件对应的文件标识;所述文件块是对文件数据集中的文件进行聚合所生成的数据块;

[0007] 对所述多个文件块的块标识进行分组,形成多个块标识组;

[0008] 对所述多个块标识组进行组间乱序,并对至少部分所述块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列;

[0009] 按照所述文件标识序列中文件标识的顺序,分批次读取与当前批次包含的文件标识相对应的文件。

[0010] 第二方面,本发明提供了一种数据读取装置,包括:

[0011] 获取模块,用于获取多个文件块的块标识,并确定所述文件块中每个文件对应的文件标识;所述文件块是对文件数据集中的文件进行聚合所生成的数据块;

[0012] 分组模块,用于对所述多个文件块的块标识进行分组,形成多个块标识组;

[0013] 乱序模块,用于对所述多个块标识组进行组间乱序,并对至少部分所述块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列;

[0014] 读取模块,用于按照所述文件标识序列中文件标识的顺序,分批次读取与当前批次包含的文件标识相对应的文件。

[0015] 第三方面,本发明提供了一种计算机设备,包括:存储器和处理器,存储器和处理器之间互相通信连接,存储器中存储有计算机指令,处理器通过执行计算机指令,从而执行上述第一方面或其对应的任一实施方式的数据读取方法。

[0016] 第四方面,本发明提供了一种计算机可读存储介质,该计算机可读存储介质上存储有计算机指令,计算机指令用于使计算机执行上述第一方面或其对应的任一实施方式的数据读取方法。

[0017] 第五方面,本发明提供了一种计算机程序产品,包括计算机程序/指令,所述计算

机程序/指令被处理器执行时实现上述第一方面或其对应的任一实施方式的数据读取方法。

[0018] 本发明基于块标识实现对多个迭代过程所需文件块的分组,形成多个块标识组,并利用块标识组实现组间乱序以及组内文件的乱序,从而可以方便快速地实现对整个文件数据集的乱序,在保证模型训练准确度的同时,可以提高乱序效率,能够保证读取效率。并且,对文件块进行分组,使得读取当前批次的文件时,这些文件更容易命中同一组的文件块,可以提高命中比例,在当前批次只需要处理少量文件块中的文件,可以进一步提高读取效率。

附图说明

[0019] 为了更清楚地说明本发明具体实施方式或相关技术中的技术方案,下面将对具体实施方式或相关技术描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图是本发明的一些实施方式,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。

[0020] 图1是根据本发明实施例的海量小文件数据集的主要操作内容的框架示意图;

[0021] 图2是根据本发明实施例的分布式系统进行数据聚合的一种过程示意图;

[0022] 图3是根据本发明实施例的聚合形成多个文件块的一种过程示意图;

[0023] 图4是根据本发明实施例的数据读取方法的流程示意图;

[0024] 图5是根据本发明实施例的训练层级的结构示意图;

[0025] 图6是根据本发明实施例的训练任务的一种工作流程示意图;

[0026] 图7是根据本发明实施例的一种训练架构示意图;

[0027] 图8是根据本发明实施例的另一数据读取方法的流程示意图;

[0028] 图9是根据本发明实施例的形成全局文件列表的一种过程示意图;

[0029] 图10是根据本发明实施例的对索引进行乱序的一种过程示意图;

[0030] 图11是根据本发明实施例的再一数据读取方法的流程示意图;

[0031] 图12是根据本发明实施例的分批次读取数据的流程示意图;

[0032] 图13是根据本发明实施例的基于文件块进行文件寻址的过程示意图;

[0033] 图14是根据本发明实施例的基于寻址信息对象实现寻址的时序图;

[0034] 图15是根据本发明实施例的基于SSD进行性能对比的测试结果图;

[0035] 图16是根据本发明实施例的基于NVME进行性能对比的测试结果图;

[0036] 图17是根据本发明实施例的基于HDD进行性能对比的测试结果图;

[0037] 图18是根据本发明实施例的数据读取装置的结构框图;

[0038] 图19是本发明实施例的计算机设备的硬件结构示意图。

具体实施方式

[0039] 为使本发明实施例的目的、技术方案和优点更加清楚,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。

[0040] 训练任务使用的数据集,基本都在远端存储系统或者云存储中,任务真正运行时,数据集必须本地化。而拉取远端数据集到本地,就需要有更好的方案,解决数据集的拉取效率、缓存、管理等存在的问题。为了解决海量数据集的处理问题,国内外开发了很多关于数据集的分布式存储系统,以聚合存储资源,存储海量数据集。为了解决海量数据集的训练问题,市场也开发了一些相关组件,解决训练任务读取海量数据的问题。

[0041] 目前,为了兼顾云原生环境,解决因云原生环境计算和存储分离,导致数据访问延迟和远程获取数据集带宽开销大等问题,市场推出的数据集处理框架有着各自的架构特点和市场定位。为了兼顾不同存储、不同数据类型文件、不同应用场景,目前都是开发成了通用型的数据处理框架,导致对千万级别、亿万级别规模的海量小文件数据集 (small file dataset) 的处理性能并不是特别好,有很多问题需要优化。例如,市场上推出的Alluxio (分布式超大规模数据编排平台)、Fluid等,体量都比较大,是通用性解决方案,对海量小文件数据集的解决方式并不是特别友好。部分数据集处理工具,主要对接对象存储系统,或者底层使用HDFS (Hadoop Distributed File System,一种分布式文件系统) 等,处理海量小文件数据的对接框架基本没有。

[0042] 在大规模计算集群中,许多训练任务都是并发运行,每个训练任务使用的数据集可能不同。训练任务给底层存储系统发出大量小文件读取请求。这种访问方式,对存储系统是一种挑战,存储系统处理小文件的效率较低,导致训练速度变慢。目前系统基本都是元数据 (meta) 密集型工作负载。

[0043] 训练用的KB级小文件,文件越小,元数据操作的耗时占比也就越高;I/O (输入/输出) 越小,延时和QPS (Queries-per-second,每秒查询率) 越超过吞吐成为主要矛盾。因此,对于小文件的乱序 (shuffle) 和读取,保证延时和QPS是提升性能的关键。

[0044] 具体地,训练用的数据集的主要操作包括读操作和写操作。写操作过程体现在:拉取远端数据集,写入本地存储中 (内存、NVME、SSD、HDD、本地搭建的共享存储系统、分布式存储系统或对象存储系统等);读操作过程体现在:训练任务运行时,按照批次 (batch),读取已拉取的数据集。但是,如果将远端存储数据集预拉取到本地搭建的存储系统,还需要记录数据集的元数据 (meta) 信息,而元数据的管理方式,影响到训练时获取训练文件列表的效率,以及乱序效率。即使本地化后,海量小文件的读取方式,也影响训练任务的读取效率,进而影响训练效率。

[0045] 针对海量小文件数据集的处理方案,一直是市场和学术科研界的痛难点问题,也没有行之有效的技术方案和开源架构,解决海量小文件数据集面临的一系列问题。针对数万目录和数千万、亿万小文件的海量小文件数据集,目前的处理方法包括:

[0046] (1) 使用共享分布式存储系统,将数据集目录挂载到各个计算节点,训练任务直接从共享存储读取数据。但是,由于高频率的元数据 (meta)、存储 (storage) 交互访问,以及高并发性和持续的I/O读取等,千万小文件,就需要超过千万次的元数据和数据流量交互,导致易饱和现有分布式文件系统。分布式存储系统处理小文件的效率,严重影响了训练效率。

[0047] (2) 以优化的数据格式封装大量小文件,虽然定制的格式可以减少元数据工作负载,但是,读取工作负载没有改变,存储系统和计算节点之间有限带宽是性能瓶颈。同时,针对封装的小文件,需要定义各自的封装和解译规则,需要在存储系统和训练之间维护一套管理系统,处理和维持读取的数据,增加了复杂性。

[0048] (3) 硬件上通过燃烧经费购买高性能硬件,通过物理手段进行规模性扩展,解决性能瓶颈问题,但这种方式成本较高。

[0049] 目前海量数据集基本都是元数据和存储数据分离的架构,而该种架构模式,有其优势的同时,也会导致海量小文件数据集的高密集型元数据交互,以及高频次的小文件I/O(输入输出)开销,会导致基于该模式设计的文件系统,在面对海量小文件时,基本没有性能可言。

[0050] 海量小文件数据集的主要操作内容可参见图1所示,其可能涉及的问题包括:读/写数据问题,元数据管理和密集型交互,数据一致性,数据缓存,数据存储,存储I/O负载等。

[0051] 如图1所示,写操作包括对文件块(chunk)及其元数据(meta)的写操作,文件块可存储在相应的存储系统中,读操作时可以读取该存储系统中的文件块,并对其进行解析;元数据可以是键值对(key-value)形式,其存储在数据库中,且还可以存储元数据快照。读操作时,也可读取海量元数据的元数据列表。并且,读操作主要涉及数据缓存管理、乱序(shuffle)、校验数据完整性、k8s调度(kubernetes,是一个用于自动化部署、扩展和管理容器化应用程序的开源系统)等。

[0052] 在使用海量小文件数据集进行AI训练时,可能面临的问题包括:

[0053] (1) 磁盘I/O瓶颈:因为每个文件都需要单独的I/O操作,导致训练读取海量小文件会存在大量的磁盘I/O开销,这可能导致数据加载和处理时间变慢,从而影响整体训练性能。

[0054] (2) 元数据开销:海量小文件通常附带额外的元数据,例如文件名、时间戳等,除了消耗存储空间,还会导致密集型的元数据交互,元数据的处理性能,会影响海量小文件的处理性能。在处理海量小数据集时,这种元数据开销会成为一个挑战,影响文件系统性能。

[0055] (3) 文件系统限制:某些文件系统对可以存储在目录或分区中的最大文件数量有限制。在处理数百万或数十亿个小文件时,这些限制可能会在文件组织和存储管理方面带来挑战。

[0056] (4) 数据预处理复杂性:预理由小文件组成的海量数据集可能需要大量的计算和时间。数据清理、增强或特征提取等操作需要有效地应用于每个文件,这可能会增加总体训练时间。

[0057] (5) 数据加载和转换:将海量小文件加载到内存中可能是内存密集型的,并且可能需要仔细的内存管理。此外,由于文件格式和结构的可变性,将数据转换为对应的训练格式可能具有挑战性。

[0058] (6) 并行化和分布式计算:使用海量小文件数据集训练模型可能需要分布式计算框架或并行处理技术来有效地处理计算负载。跨多个节点协调数据分发和同步可能很复杂。

[0059] 针对海量数据集,通常会将海量小文件加载到内存或分布式存储系统中,以便后续处理和训练,是目前常用的方式。一般会使用预加载、预处理手段将远端数据集加载到本地搭建的文件系统,也可以在训练时按需加载,但涉及到数据预拉取、数据流式传输等技术,需要优化数据加载的效率和速度。对于海量小文件,都存在加载效率低和加载性能差的问题。

[0060] AI训练任务使用的海量小文件数据集,一般都是分类整理好的,且都是读写分离

场景,为了提高效率,基本是拉取一次,读访问频繁的场景。而且,AI训练任务,一般要经过多次迭代(epoch),每次迭代都会读取全部数据集,但是,每次迭代每个文件也只读取一次。为了解决AI训练任务中海量小文件数据集的至少部分相关问题,针对海量小文件拉取一次,读多次的特性,本发明构建分布式系统,并在AI训练任务的读取数据集阶段,利用训练库与该分布式系统交互,实现训练数据的加速读取,缩短训练时间,提高训练效率,解决海量小文件数据集的读取效率和性能等相关问题。

[0061] 其中,本实施例中采用分布式系统存储文件数据集及其相应的元数据;例如,该分布式系统可以为CacheFS分布式系统。该分布式系统可以从远端数据源拉取并聚合文件数据集,并将拉取到的文件数据集进行聚合存储,使得后溪可以从该分布式系统中读取聚合的文件数据集作为训练数据。

[0062] 图2示出了该分布式系统进行数据聚合的一种过程示意图。如图2所示,该分布式系统包括服务端(例如,CacheFS Server)以及设置在计算节点的客户端(例如,CacheFS Client)。并且,该分布式系统可以将各个计算节点的存储单元进行聚合,将多个计算节点(多个计算节点形成计算集群)的多个磁盘等存储单元聚合成一个存储空间,对接到分布式系统中,可以防止因海量的文件数据集过大,而导致单个计算节点无法缓存完成远端数据源中数据的问题。其中,该存储单元例如可以是NVMe(非易失性内存主机控制器接口规范)、SSD(固态硬盘)、HDD(硬盘驱动器,即传统的机械硬盘)等。

[0063] 如图2所示,分布式系统的客户端可以向远端数据源发起拉取请求,远端数据源响应该拉取请求,从而可以向分布式系统下发文件数据集。具体地,该客户端可以采用多通道多进程的方式工作,扫描远端数据源中的文件数据集,并保持该文件数据集原始的相对目录结构,按照聚合规则,无损聚合为文件块(chunk),并将文件块编号处理,形成每个文件块的ID,即chunk ID。

[0064] 客户端可以使用fuse挂载指定路径到各个计算节点,对挂载路径下的文件数据集进行读、写、修改、删除等操作。因为客户端对接的聚合存储中的数据,是聚合后的文件块,并不是原始的文件数据集的目录文件层级结构,所以,客户端还可以具备数据视图等功能,使得用户看到的文件数据集层级结构和原始的目录文件层级类似。同时,在训练时,为了提高读取效率,可以预缓存文件数据集到计算节点本地的存储单元(例如,内存、NVMe、SSD、HDD等)中,具备本地缓存文件数据集、管理等功能,也具有元数据上报等功能。

[0065] 如图2所示,聚合生成的文件块,写入至聚合存储,即存储到计算集群聚合的存储空间;并且,将文件块的元数据上报至服务节点。存储完成后,客户端可以上报存储成功的文件块的先关信息。

[0066] 服务节点的服务端主要记录聚合的文件块的元数据、同步元数据等;其中,该元数据具体可以包括:文件块的ID、文件块所聚合的文件名列表及相对目录结构、文件块所聚合的文件总数、文件块大小、文件块创建/修改时间戳等。服务节点的数据库可以是redis或mysql等,能够记录聚合的存储信息、各客户端的相关信息等。

[0067] 图3示出了将文件数据集聚合成多个文件块的一种过程示意图。如图3所示,文件数据集中包含多个文件;例如,该文件数据集为海量小文件的数据集,其中的文件为小文件。可以从该文件数据集中抽取能够聚合的多个文件,形成一定大小的文件块;如图3所示,可以将4KB的文件1、128KB的文件2以及其他多个文件(图3中未示出)聚合为一个文件块,即

文件块1,例如,该文件块1大小为4MB;类似地,也可以聚合形成其他的文件块2。

[0068] 这些文件块可以存储在计算集群聚合形成的存储空间中,基于聚合存储的方式存储所有的文件块,共m个文件块。如图3所示,文件块1中包含文件1、文件2、…、文件n等,从而实现以文件块(chunk)的形式对小文件等数据的存储。

[0069] 并且,在聚合形成文件块的同时,可以记录该文件块所包含的文件列表的元数据,并将元数据上传至服务节点。

[0070] 由于海量小文件的文件数据集过大,计算节点本地存储单元可能无法缓存完成本次迭代(epoch)训练的数据集,从而导致出现边删除边拉取的问题;由于删除和拉取都需要耗时,导致训练任务等待,进而耗时增加,影响训练效率。此外,训练过程一般需要多次迭代,即使使用分布式方式,每个计算节点训练完成,基本也需要拉取一遍完整的文件数据集,如果存储空间不充足,会导致本次迭代训练删除的内容,下次迭代还需要重新拉取,严重影响训练效率。

[0071] 本实施例中的分布式系统,可以实现将远端海量小文件的文件数据集,预拉取或任务触发拉取到分布式系统聚合的存储空间中,可以一次拉取,多次读取。每个数据文件,只需和远端数据源交互一次即可,以后多次迭代训练的过程中,各计算节点只需与本地挂载的客户端交互就行。任务训练完成后,客户端可以设置随任务结束,清理拉取的文件数据集,也可以设置定时任务,定期清理超过预设时长不使用的文件数据集等,实现数据集管理功能。

[0072] 根据本发明实施例,提供了一种数据读取方法实施例,需要说明的是,在附图的流程图示出的步骤可以在诸如一组计算机可执行指令的计算机系统中执行,并且,虽然在流程图中示出了逻辑顺序,但是在某些情况下,可以以不同于此处的顺序执行所示出或描述的步骤。

[0073] 在本实施例中提供了一种数据读取方法,可用于上述的计算节点,图4是根据本发明实施例的数据读取方法的流程图,如图4所示,该流程包括以下步骤。

[0074] 步骤S401,获取多个文件块的块标识,并确定文件块中每个文件对应的文件标识;该文件块是对文件数据集中的文件进行聚合所生成的数据块。

[0075] 本实施例中,如上所述,从远端数据源拉取的文件数据集,可以被聚合为多个文件块(chunk),这些文件块可以存储在聚合存储的存储空间中;并且,该文件块的元数据(meta)可以存储至服务节点,例如存储在服务节点的数据库中。在训练任务的读取数据集阶段,需要读取这些文件块,此时可以获取这些文件块的块标识。一般情况下,该训练任务需要读取的是整个文件数据集的文件块。

[0076] 其中,该块标识是唯一表示文件块的标识;例如,该块标识具体可以是文件块的ID。该文件标识时唯一表示相应文件的标识;例如,该文件标识具体可以是文件的ID。其中,可以从用于存储元数据的数据中心,提取出所有文件块以及其中文件的元数据,该数据中心例如可以是图2所示的服务节点的数据库,基于获取到的元数据确定相应的块标识和文件标识。

[0077] 训练任务(例如,AI训练任务)一般分为多次迭代(epoch),每次迭代又分为多个批次,分批次读取所需的文件。其中,在训练任务开始时,首先需要获取文件数据集的元数据,其包括该文件数据集的全部文件列表,以及用于训练的标签信息,例如分类标记信息等。在

每次迭代训练时,为了提高模型训练的准确性,一般需要对文件数据集的全部文件进行乱序(shuffle),这就导致读取海量文件时的效率较低。例如,迭代读取小文件时,现有的训练库,例如pytorch/tensorflow等官方训练库,都会打开每个小文件,读取文件数据,不仅耗时长,而且效率低。

[0078] 本实施例中,构建自身所需的训练库,以加速AI训练中的海量小文件数据集读取效率,减少小文件读取时,与存储系统的交互次数,降低频繁的I/O开销,解决训练中海量小文件数据集的相关痛点问题。图5为本实施例提供的训练层级的结构示意图。如图5所示,有训练库指示客户端,从分布式系统中读取相应的数据,以完成应用程序(Application)所需的训练任务。

[0079] 具体地,在进行训练任务时,可以使用训练库,根据指定的文件数据集的相关路径信息,与存储元数据的数据中心进行交互,以从该数据中心提取出文件数据集的元数据,具体可以包括文件块的元数据以及其中文件的元数据。例如,该文件块的元数据具体可以包括:文件块的ID(chunk Id)、文件块名(chunk name)、相应的索引节点(inode)、文件块大小等;该文件的元数据包括文件名、文件ID等。

[0080] 根据这些元数据,可以生成文件块的块标识和文件的文件标识。例如,将文件块的ID作为相应的块标识,将文件的ID作为相应的文件标识。

[0081] 可以理解,文件块的元数据,与文件块所聚合的文件,二者是分开存储的;如图2所示,该文件块的元数据存储在服务节点,而文件块本身存储在聚合存储的存储空间。因此,本实施例中,在读取文件块时,也需要分开读取其元数据以及相应的文件块,完成训练任务的工作流程示意图可参见图6。

[0082] 如图6所示,基于训练任务可以触发训练读取,该训练任务可以由应用程序确定,例如pytorch/tensorflow训练任务等。之后,由训练库从服务节点读取相应的元数据,并基于客户端读取相应的文件块。

[0083] 步骤S402,对多个文件块的块标识进行分组,形成多个块标识组。

[0084] 本实施例中,如上所述,为了提高模型训练的准确性,一般需要对文件数据集的全部文件进行乱序(shuffle)。由于文件数据集包含海量的文件,其文件数量一般至少百万级以上,若直接对文件数据集进行乱序,则会存在乱序效率较低的问题。本实施例中,先获取文件数据集的元数据,基于该元数据确定的块标识和文件标识进行乱序,可以降低乱序过程的处理量。

[0085] 其中,可以基于训练库实现乱序。图7示出了本实施例中的训练架构示意图,如图7所示,训练库从服务节点获取到元数据后,基于该元数据进行乱序,进而可以基于乱序结果,指示客户端从聚合存储中读取相应的文件块,并缓存至本地缓存中,该本地缓存具体可以是本地内存,或者本地磁盘,例如NVMe、SSD等。其中,可以由数据加载器(DataLoader)实现对文件块的加载。

[0086] 具体地,在进行乱序操作时,首先对多个文件块进行分组;其中,通过对多个文件块的块标识进行分组,可以表示对文件块的文组,并且,将每一组块标识称为块标识组。可以理解,该块标识组中包含多个块标识。

[0087] 例如,该块标识是块标识的ID,若每四个文件块归为一组,则可将相邻的四个块标识ID分为一组,形成包含四个块标识ID的块标识组。

[0088] 步骤S403,对多个块标识组进行组间乱序,并对至少部分块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列。

[0089] 本实施例中,为了能够实现对文件数据集的乱序,可以对多个块标识组进行组间乱序,即以块标识组为单位,在块标识组之间进行乱序;例如,将文件数据集分为四个块标识组:块标识组1、块标识组2、块标识组3、块标识组4,可以对四个块标识组进行组间乱序,乱序后的结果例如可以为:块标识组3、块标识组1、块标识组4、块标识组2等。

[0090] 并且,对于每一块标识组,其对应多个文件,故其包含多个文件的文件标识。本实施例中,还对块标识组中的文件标识进行组内乱序,其中,部分或全部的块标识组均可进行组内乱序。例如,若块标识组1包含三个文件的文件标识,且分别为:文件标识1-1、文件标识1-2、文件标识1-3,则可单独对这三个文件标识进行组内乱序,乱序后的结果例如可以是:文件标识1-2、文件标识1-3、文件标识1-1;对于其他块标识组,也可按照类似的方式进行组内乱序,此处不作详述。

[0091] 可以理解,可以先进行组间乱序、再进行组内乱序,也可以先进行组内乱序、再进行组间乱序,此处不作限定。

[0092] 本实施例中,整个文件数据集包含多个文件,其对应一定的文件列表;而每个文件具有特定的文件标识,故该文件数据集也具有原始的文件标识列表。在进行组内乱序和组间乱序后,可以实现对该文件标识列表的乱序;为方便描述,将乱序后的文件标识列表称为文件标识序列。

[0093] 由于文件数据集包含海量的文件,相应地,其所对应的文件标识的数量也是海量的,本实施例采用组内乱序+组间乱序的方式,可以更简单地实现对整个文件数据集的乱序。

[0094] 例如,将文件数据集的所有文件块分为 m 组,形成 m 个文件块组,每个文件块组对应一个块标识组;并且,每个文件块组包含 n 个文件,即每个块标识组包含 n 个文件标识。可以理解,该文件数据集包括 $m \times n$ 个文件。本实施例中,在进行组间乱序时,是对 m 个元素(即 m 个块标识组)进行乱序;在进行组内乱序时,是对 n 个元素(即 n 个文件标识)进行乱序,可以极大降低乱序元素的基数,从而可以更简单、更快速地完成乱序。

[0095] 步骤S404,按照文件标识序列中文件标识的顺序,分批次读取与当前批次包含的文件标识相对应的文件。

[0096] 本实施例中,该文件标识序列所包含的只是文件标识,其并不包含文件本身,但是,基于该文件标识可以确定文件数据集中相应的文件,进而可以对该文件进行读取。具体地,可以读取相应的文件块,该过程可参见图6和图7所示。

[0097] 分批次读取的文件,可以用于模型训练,例如训练识别模型、分类模型等,该文件具体可以为图像数据、文本数据等。

[0098] 其中,在一次迭代过程中,需要分批次(batch)对文件数据集中的文件进行读取,直至读取完该文件数据集;例如,可以基于预设的批次大小(batch size),在每一批次读取该批次大小的文件。在当前批次,可以确定当前批次包含哪些文件标识,进而基于这些文件标识读取相应的文件,完成对当前批次文件的读取。

[0099] 在下一迭代时,重复分批次读取文件数据集中文件的过程,通过多次迭代,完成训练。

[0100] 本实施例中,由于乱序过程是在对文件块分组的基础上进行的,使得在文件标识序列中,属于同一块标识组的多个文件标识,其只属于少量的文件块;进而,在分批次读取文件时,当前批次对应的文件标识,也只会对应少量的文件块,使得再当前批次只需要处理少量文件块中的文件。

[0101] 例如,若文件数据集包括文件块1、文件块2、文件块3和文件块4,且文件块1、文件块2为一个文件块组,文件块3、文件块4为一个文件块组。若训练过程中的批次大小小于该文件块组所包含的文件数量,则对于某一批次的文件标识,大概率只对应其中一个文件块组内的文件,或者说,对于某一批次的文件标识,其中的大部分文件标识大概率均对应同一文件块组;例如,当前批次的文件标识,只对应第一个文件块组,故此时只需要处理文件块1和文件块2中的数据,从而可以减小需要处理的数据量;若当前批次的文件标识还对应第二个文件块组,则还需要对文件块3和文件块4进行处理,例如需要对文件块3和文件块4进行查询等。

[0102] 可以理解,该批次大小可以小于文件块组内的文件数量,即批次大小可以小于块标识组中的文件标识数量,也可以大于块标识组中的文件标识数量,本实施例对此不作限定。若批次大小大于块标识组中的文件标识数量,当前批次的文件标识对应的文件块虽然属于多个文件块组,但所对应的文件块组的数量仍然是比较少的,仍然可以保证当前批次的文件标识只命中少量的文件块,进而方便后续基于命中的文件块实现文件读取。

[0103] 本实施例提供的数据读取方法,基于块标识实现对多个迭代过程所需文件块的分组,形成多个块标识组,并利用块标识组实现组间乱序以及组内文件的乱序,从而可以方便快速地实现对整个文件数据集的乱序,在保证模型训练准确度的同时,可以提高乱序效率,能够保证读取效率。并且,对文件块进行分组,使得读取当前批次的文件时,这些文件更容易命中同一组的文件块,可以提高命中比例,在当前批次只需要处理少量文件块中的文件,可以进一步提高读取效率。

[0104] 在本实施例中提供了一种数据读取方法,可用于上述的计算节点,图8是根据本发明实施例的数据读取方法的流程图,如图8所示,该流程包括以下步骤。

[0105] 步骤S801,获取多个文件块的块标识,并确定文件块中每个文件对应的文件标识;文件块是对文件数据集中的文件进行聚合所生成的数据块。

[0106] 详细请参见图4所示实施例的步骤S401,在此不再赘述。

[0107] 步骤S802,对多个文件块的块标识进行分组,形成多个块标识组。

[0108] 详细请参见图4所示实施例的步骤S402,在此不再赘述。

[0109] 在一些可选的实施方式中,上述步骤S802“对多个文件块的块标识进行分组,形成多个块标识组”,具体可以包括以下步骤A1至步骤A2。

[0110] 步骤A1,根据文件块中的文件数量,确定分组大小;文件块中的文件数量与分组大小之间为负相关关系。

[0111] 本实施例中,聚合形成的文件块,其包含多个文件;例如,该文件数据集为小文件数据集(small file dataset),假设每个小文件的大小为1KB,所聚合形成的文件块大小为4MB,则该文件块共包含4000个小文件,其文件数量即为4000。

[0112] 在对文件块进行分组时,每个文件块组包含多个文件块,故文件块的文件数量越大,该文件块组内的文件数量也越大;同样地,若文件块组的分组大小(group size)越大,

该文件块组内的文件数量也越大;其中,该分组大小表示文件块组所包含的文件块的数量。

[0113] 若文件块组内的文件数量较大,则会导致按照该分组大小所分成的块标识组中的文件标识数量也较大;由于块标识组中的文件标识是乱序的,这样的话,在当前批次需要读取的文件的文件标识,可能对应较多的文件块,这不利于后续处理。因此,本实施例中,将文件块中的文件数量与分组大小之间设为负相关关系,即文件块中的文件数量越多,所设置的分组大小越小,以能够自适应设置分组大小。合适的分组大小,可以保证当前批次命中同一文件块组中文件的命中率,从而可以保证当前批次基于文件块的文件读取效率。

[0114] 可选地,在聚合形成文件块时,可以预先设置该文件块固定的大小,在该文件块大小的条件下对多个文件进行聚合,该文件块所聚合的文件数量与每个文件的大小有关。例如,文件块大小为4MB,则该文件块所聚合的所有文件的大小之和不能超过4MB。并且,在设置分组大小时,也可以根据文件数据集的规模进行设置,即根据文件数据集所包含文件数量(或者文件块数量)进行设置;该文件数据集的规模越大,该分组大小也可以越大。

[0115] 或者,由于每一批次处理的文件数量,一般是2的指数次,即批次大小(batch size)是2的指数次,例如64、128等。因此,在对文件聚合以形成文件块时,可以不限文件块的大小,而是将2的m次方(即 2^m)个文件聚合为一个文件块,即该文件块中的文件数量为 2^m ;并且,该分组大小设为2的n次方(即 2^n),其中,m和n均为整数,例如,m、n均大于1。假设每一批次需要处理的文件数量为 2^p ,即批次大小为 2^p ,则可使得当前批次需要处理的文件有更大概率属于同一文件块组。例如,若p小于或等于m+n,则当前批次需要处理的文件,一定属于同一文件块组,从而可以保证后续的文件读取效率。

[0116] 步骤A2,按照分组大小对多个文件块的块标识进行分组,形成多个块标识组。

[0117] 本实施例中,在确定该分组大小后,即可按照该分组大小对文件块的块标识进行分组,形成该分组大小的块标识组,即该块标识组包含该分组大小的块标识。

[0118] 步骤S803,对多个块标识组进行组间乱序,并对至少部分块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列。

[0119] 其中,实现乱序的文件标识可以为相应文件的ID,但文件ID本身比较复杂,例如某图片数据集中的某个文件的ID为“ILSVRC2012_test_00000001.JPEG”,甚至,文件ID还可能基于相应的文件块名进行表示,即文件ID包含所属文件块的信息,例如某文件ID为:“imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000001.JPEG”。因此,直接以文件ID进行乱序,实现比较复杂,且存储乱序后的文件标识序列也需要较大的存储空间。

[0120] 本实施例中,以文件在整个文件数据集中的索引,作为该文件的文件标识,即该文件标识为相对应的文件在多个文件块所包含的所有文件中的索引。

[0121] 具体地,在开始训练任务时,可以生成对应文件数据集所有文件的全局文件列表,该全局文件列表能够表示文件数据集中的文件与其元数据之间的映射关系,进而在读取文件时可以基于该全局文件列表定位到相应的文件;并且,利用该全局文件列表还可以确定每个文件在该全局文件列表中的位置,将该位置作为文件的索引。其中,该全局文件列表具体可以是文件数据集所有文件的描述数据的列表,该描述数据具体可以是相应文件的元数据;可以理解,该全局文件列表中元素的排列顺序,也可表示文件数据集所有文件的一种排列顺序。

[0122] 可选地,利用该全局文件列表确定文件索引的过程可以包括以下步骤B1至步骤

B3。

[0123] 步骤B1,在模型训练的第一次迭代过程中,以块标识组为单位生成包含多个文件的描述数据的文件列表。

[0124] 步骤B2,对文件列表中的描述数据进行乱序,并将乱序后的文件列表按照块标识组的顺序,形成包括多个文件列表中描述数据的全局文件列表。

[0125] 步骤B3,将文件的描述数据在全局文件列表中的位置下标作为相应文件的索引。

[0126] 本实施例中,在第一次迭代(epoch)过程中,可以以文件列表为单位进行乱序,不仅可以形成乱序后的全局文件列表,还可以使用乱序后的文件进行后续训练,保证第一次迭代训练的准确性。

[0127] 例如,以文件块ID表示块标识,在开始训练任务时,可以生成包括所有文件块ID的文件块ID列表,根据文件块ID的数量以及指定的线程数量(默认50),可以自适应切分该文件块ID列表,切分得到的每个子列表的大小=文件块ID的数量/线程数量,并向上取整。开启该指定线程数量的线程,每个线程批量查询各自子列表中的文件块的聚合信息,并将各自文件块聚合的小文件解压缩出来,与chunk Id、chunk name等元数据进行映射,形成小文件与chunk Id、chunk name的映射关系。

[0128] 在第一次迭代时,将该文件块ID列表按照合适的分组大小(group size)进行分组,形成多个块标识组。分组后,再将各块标识组中,每个文件块(chunk)聚合的小文件,以块标识组为单位生成相应的文件列表(group_files),并对各块标识组的文件列表进行乱序(shuffle);之后,将各块标识组乱序后的文件列表,按照块标识组的顺序,放入全局文件列表中,形成包含所有文件的全局文件列表。

[0129] 图9示出了形成全局文件列表的一种过程示意图。如图9所示,该文件数据集包括6个文件块,分别为:文件块1、文件块2、文件块3、文件块4、文件块5、文件块6。对这6个进行分组,可以形成相应的文件块组。如图9所示,可以先以文件块为单位,对所有的文件块进行乱序,图9中的乱序结果为:文件块4、文件块1、文件块6、文件块2、文件块5、文件块3。若每三个文件块作为一组,则可形成包括文件块4、文件块1、文件块6的文件块组A,以及包括文件块2、文件块5、文件块3的文件块组B。

[0130] 并且,每个文件块组包含多个按序排列的文件,使得每个文件块组对应相应的文件列表。如图9所示,文件块4包括文件1、文件2、文件3、文件4,文件块1包括文件5、文件6、文件7、文件8,以此类推。因此,文件块组A对应的文件列表包括从文件1至文件12,文件块组B对应的文件列表包括从文件13至文件24。

[0131] 之后,对每个文件块组的文件列表内的文件分别进行乱序,如图9所示,对于文件块组A,其乱序后的文件列表为:文件5、文件2、文件11、文件9、文件1、文件8、文件12、文件4、文件7、文件3、文件6、文件10;对于文件块组B,其乱序后的文件列表为:文件21、文件17、文件13、文件18、文件22、文件15、文件14、文件20、文件24、文件16、文件23、文件19。将这两个乱序后的文件列表按照先文件块组A、后文件块组B的顺序,可以拼接形成完整的全局文件列表,该全局文件列表具体可参见图9的最后一行所示。

[0132] 在确定全局文件列表后,即可基于该全局文件列表确定每个文件的索引。其中,文件在该全局文件列表中的位置,可以用位置下标表示,相应地,该位置下标可以作为文件索引。

[0133] 例如,若各块标识组的文件列表长度为len,则对于每一块标识组所对应的文件,可生成从idx到idx + len的索引列表,该索引idx可以从0开始,将每个索引列表按序排列,即可形成全局索引列表,该全局索引列表即可表示每个文件的索引。其中,该全局索引列表中的元素值idx,就是全局文件列表中对应文件的下标samples[idx],因此,通过samples[idx]可以获取全局文件列表中对应位置的文件信息;其中,samples表示全局文件列表。

[0134] 并且,参见图8所示,上述步骤S803“对多个块标识组进行组间乱序,并对至少部分块标识组中的文件标识进行组内乱序”可以包括以下步骤S8031至步骤S8032。

[0135] 步骤S8031,构建二维数组;该二维数组中的第一维元素表示相应的块标识组,第一维元素对应的第二维元素表示相应块标识组中的文件标识。

[0136] 其中,在第一次迭代过程中,由于该全局文件列表已经过乱序,故可以直接基于该全局文件列表进行文件读取,即分批次读取相应的文件。例如,若批次大小(batch size)为4,参见图9所示,在第一批可以读取文件5、文件2、文件11、文件9,在下一批次,即第二批,可以读取文件1、文件8、文件12、文件4,以此类推。

[0137] 本实施例中,为便于对文件数据集进行乱序,特别是从第二次迭代开始,需要在每一迭代过程均进行乱序,本实施例基于文件的索引构建二维数据。

[0138] 具体地,将块标识组对应二维数组的第一维元素,将每个块标识组中的文件标识(即索引)作为相应第一维元素内的第二维元素,从而可以形成包含所有文件索引的二维数组。

[0139] 例如,该二维数组为:{{idx₁₁, idx₁₂, ..., idx_{1n}}, {idx₂₁, idx₂₂, ..., idx_{2n}}, ..., {idx_{m1}, idx_{m2}, ..., idx_{mn}}};其中,idx_{ij}表示第i个块标识组中的第j个索引。对于该二维数组中的第一个第一维元素{idx₁₁, idx₁₂, ..., idx_{1n}},其中的每个第二维元素(例如,idx₁₁、idx₁₂等)均表示属于相应块标识组的索引。其中,该索引idx_{ij}本质上是全局文件列表中的某个索引,以i和j进行表示,只是为了便于表示该索引所属的块标识组、以及其对应应该块标识组内的哪一位置。

[0140] 步骤S8032,对二维数组中的多个第一维元素进行乱序,并对至少部分第一维元素对应的第二维元素进行乱序。

[0141] 本实施例中,对二维数组中的多个第一维元素进行乱序,相当于对块标识组进行组间乱序;对第一维元素对应的第二维元素进行乱序,相当于对块标识组中的文件标识进行组内乱序。因此,通过对该二维数据进行两个维度的乱序,可以很方便地实现对所有文件的文件标识进行组内乱序和组间乱序。

[0142] 步骤S8033,按照二维数组中元素乱序后的顺序,将乱序后的二维数组转换为一维的全局索引列表;该全局索引列表作为文件标识序列。

[0143] 可以理解,该二维数组中第一维元素的顺序,就是文件块的分组顺序,即块标识组的顺序;并且,第一维元素对应的第二维元素的顺序,就是各块标识组中文件列表顺序,即文件标识的顺序。在对二维数组进行乱序后,保持其中元素乱序后的顺序,将二维数组转为一维的列表,该列表即为包含所有索引的全局索引列表,该全局索引列表即为上述的文件标识序列。

[0144] 例如,在第二次迭代时,可以基于第一次迭代所确定的全局文件列表确定每个文件的索引,进而可以基于这些文件的索引生成相应的二维数组,并将该二维数组中的各一

维数组(即第一维元素)乱序,即相当于将块标识组进行乱序,但是各块标识组中的文件顺序保持不变;并且,对二维数组中的每个一维数组内的元素(即第二维元素)分别进行乱序,即相当于将各块标识组中的文件乱序。可以理解,基于该二维数据进行乱序,是通过下标索引将文件间接乱序,并不需要直接对海量的文件进行乱序。在乱序后,将乱序的二维数组转换为列表,即可生成本次迭代所需的全局索引列表,该全局索引列表中的元素均是索引,并不是具体的文件信息。之后,即可按照批次大小,按照顺序从全局索引列表中分批次返回该批次大小数量的文件。

[0145] 图10示出了第二次迭代时对索引进行乱序的一种过程示意图。如图10所示,基于图9所示的全局文件列表,可以确定每个文件对应的索引;图10中以索引从1开始,可以依次确定索引1、索引2、…、索引24;并且,二维数组中的第一个一维数组包含索引1至索引12,其对应文件块组A,第二个一维数组包含索引13至索引24,其对应文件块组B。可以先对二维数组的一维数组进行乱序,即组间乱序,如图10所示,经过组间乱序,可以改变文件块组的顺序,即可以改变块标识组的顺序;之后再对每个一维数组中的元素进行乱序,即组内乱序,此时可以改变每一块标识组内索引的顺序,相当于改变了文件组内文件的顺序,最终形成乱序后的全局索引列表,即第二次迭代所需的全局索引列表,该全局索引列表具体可参见图10的最后一行。

[0146] 并且,在第二次迭代的每一批次,也可按照相应的批次大小,分批次读取相应索引对应的文件。如图10所示,若第一批次读取索引23、索引17、索引22、索引13,由于索引23对应文件23、索引17对应文件22、索引22对应文件16、索引13对应文件21,故可以确定,在第一批次需要读取文件23、文件22、文件16、文件21。

[0147] 本实施例中,以索引表示文件标识,并利用二维数组可以简单快速地对文件的组内乱序和组间乱序,以保持文件块组之间的顺序,也可以保持乱序后文件块组内文件的顺序,能够保证后续的读取效率。并且,这种乱序方式,可以提高每个批次命中同一组文件块的命中率,即每个批次命中的大部分文件属于同一文件块组。

[0148] 步骤S804,按照文件标识序列中文件标识的顺序,分批次读取与当前批次包含的文件标识相对应的文件。

[0149] 详细请参见图4所示实施例的步骤S404,在此不再赘述。

[0150] 本实施例提供的数据读取方法,利用二维数组可以方便快速地对整个文件数据集的乱序,在保证模型训练准确度的同时,可以提高乱序效率,能够保证读取效率。

[0151] 在本实施例中提供了一种数据读取方法,可用于上述的计算节点,图11是根据本发明实施例的数据读取方法的流程图,如图11所示,该流程包括以下步骤。

[0152] 步骤S1101,获取多个文件块的块标识,并确定文件块中每个文件对应的文件标识;文件块是对文件数据集中的文件进行聚合所生成的数据块。

[0153] 详细请参见图4所示实施例的步骤S401,在此不再赘述。

[0154] 步骤S1102,对多个文件块的块标识进行分组,形成多个块标识组。

[0155] 详细请参见图4所示实施例的步骤S402,在此不再赘述。

[0156] 步骤S1103,对多个块标识组进行组间乱序,并对至少部分块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列。

[0157] 详细请参见图4所示实施例的步骤S403,在此不再赘述。

[0158] 步骤S1104,按照文件标识序列中文件标识的顺序,分批次对当前批次包含的文件标识相对应的文件进行寻址,确定相应的寻址信息。

[0159] 本实施例中,由于文件标识并不是文件本身,且文件存储在聚合的存储空间中,为了能够从聚合的存储空间中定位到所需的文件,需要基于文件标识进行寻址,以确定所需文件的寻址信息。在确定该寻址信息后,即可从聚合的存储空间中读取到相应的文件。

[0160] 在一些可选的实施方式中,可以以文件块为单位进行寻址,具体地,上述步骤S1104“对当前批次包含的文件标识相对应的文件进行寻址,确定相应的寻址信息”可以包括以下步骤C1至步骤C3。

[0161] 步骤C1,根据当前批次包含的文件标识确定当前批次命中的目标文件所属的文件块。

[0162] 本实施例中,当前批次包含的文件标识,是当前批次需要读取的文件的文件标识,为方便描述,将当前批次需要读取的文件称为目标文件,该目标文件也是当前批次命中的文件;该目标文件所属的文件块也可称为目标文件块。

[0163] 例如,若文件标识为索引,则可以根据当前批次需要读取的文件索引,确定该文件索引所映射的文件(即目标文件),以及所映射的文件块。如图10所示,在第二次迭代的第一批次中,对于索引23,其与文件23具有映射关系,相应地,索引23映射该文件23所属的文件块,如图9所示,为文件块3;而对于索引17,其与文件22具有映射关系,相应地,如图9所示,该索引17也与文件块3具有映射关系。

[0164] 步骤C2,将属于同一文件块的目标文件归为一组,生成包含至少一个目标文件所对应文件标识的目标文件列表。

[0165] 例如,如图10所示,在第二次迭代的第一批次中,索引23、索引17、索引22、索引12分别与文件23、文件22、文件16、文件21具有映射关系,并且,文件23、文件22、文件21均属于文件块3,而文件16属于文件块2,故可以将文件23、文件22、文件21所对应的文件标识,即索引23、索引17、索引12归为一组,形成包含这三个索引的目标文件列表;并且,将文件16所对应的索引22归为另一组,形成包含该索引22的另一个目标文件列表。

[0166] 步骤C3,以目标文件列表为单位,分别确定每个目标文件列表所对应的目标文件的寻址信息。

[0167] 本实施例中,由于在每一批次只会命中少量的文件块,将属于同一文件块的目标文件归为一组,使得所生成的每个目标文件列表只对应一个文件块;以目标文件列表为单位,只需要确定一个文件块中文件的寻址信息。

[0168] 可选地,该方法还可以包括:开启多个工作进程,基于工作进程读取相应批次的文件;不同的工作进程用于读取不同批次的文件,且在工作进程读取完当前批次的文件后,工作进程用于读取下一批次的文件,直至读取完文件数据集。

[0169] 并且,在当前批次的文件读取过程中,开启与命中的文件块数量相匹配的工作线程,每个工作线程分别用于确定相应文件块中文件的寻址信息,并读取相应的文件。通过对每个文件块分别开启各自的线程,以多线程并行处理的方式进行寻址,可以提高寻址效率。

[0170] 本实施例中,由于每次迭代需要经过多个批次的处理,故可以开启多个工作进程,并行处理不同批次的文件;并且,在每次迭代过程中,还可以开启多个进程,由每个进程分别处理相应批次的文件。在处理完毕后,再基于相应的工作进程处理之后其他批次的文件;

这种并行处理的方式,也可以提高分批次处理数据的效率。

[0171] 图12示出了基于训练库实现分批次读取数据的流程示意图。如图12所示,可以从存储有海量小文件数据集的远端数据源获取训练所需的文件数据集,图12与图片数据集为例示出;在获取到该图片数据集之后,即可基于数据加载器(DataLoader)开启多个线程(即工作线程),从而能够并行地分批次迭代。并且,对于每个线程,也可开启多进程(即工作进程)。如图12所示,在每一批次的处理过程中,可以确定当前批次的索引所映射的文件(即目标文件),以及映射的文件块,进而可以开启文件块数量的进程。以图10所示的第二次迭代的第一批次为例,其最多涉及文件块组B中的三个文件块,即文件块2、文件块5、文件块3,故可以开启三个进程,分别处理文件块2、文件块5、文件块3的文件寻址过程,以确定当前批次所需的寻址信息。在确定寻址信息后,即可基于读取函数open()从分布式文件系统中读取相应的文件,并转换(transform)为所需的格式,以便于后续训练。

[0172] 其中,在图10所示的第二次迭代的第一批次,基于文件块进行文件寻址的过程可参见图13所示。例如,索引23对应文件块3中的文件23,若开启的进程3用于处理文件块3的相关数据,则可使用该进程3,根据该索引23对应的寻址信息,从分布式文件系统中读取相应的文件13。

[0173] 在一些可选的实施方式中,上述步骤S1104“对当前批次包含的文件标识相对应的文件进行寻址,确定相应的寻址信息”可以包括以下步骤D1。

[0174] 步骤D1,确定当前批次包含的文件标识命中的目标文件,并生成目标文件的寻址信息对象;寻址信息对象包括目标文件在相应文件块中的位置信息。

[0175] 其中,若基于文件的元数据等信息进行寻址,由于寻址是一般需要遍历整个文件块,例如使用for循环等方式一个文件一个文件地进行遍历,效率较低,特别是还可能涉及文件增删改、校验等操作,在数据量庞大的情况下也会影响寻址效率。本实施例中,为每个目标文件设置用于寻址的对象(object),即寻址信息对象,基于该寻址信息对象实现对目标文件的寻址。

[0176] 具体地,该寻址信息对象只需要包括读取目标文件时所需的位置信息,该位置信息具体可以包括目标文件在文件块中的句柄(file handle)、位置、偏移量(offset)等。本实施例中,采用轻量级的寻址信息对象进行寻址,寻址效率高。

[0177] 在一些可选的实施方式中,上述步骤D1“生成目标文件的寻址信息对象”具体可以包括以下步骤D11或步骤D12。

[0178] 步骤D11,以批量寻址的方式,批量生成多个目标文件的寻址信息对象。

[0179] 步骤D12,根据构建的全局字典进行寻址,确定多个目标文件的寻址信息对象;全局字典用于维护目标文件所在的文件块的所有文件的寻址信息对象。

[0180] 本实施例中,在确定寻址信息对象时,可以根据实际情况选取合适的方式;其中,可供选择的方式包括:批量寻址的方式,以及基于全局字典寻址的方式。

[0181] 可选地,将当前批次命中的目标文件所属的文件块称为目标文件块,可以基于目标文件块中文件的数量等选取合适的寻址方式。具体地,上述步骤D1“生成目标文件的寻址信息对象”还可以包括以下步骤D13。

[0182] 步骤D13,确定当前批次包含的文件标识命中的目标文件块中文件的第一数量,并确定当前批次包含的文件标识命中的、属于目标文件块的目标文件的第二数量。

[0183] 其中,设置第一数量的阈值,即第一阈值,以及第二数量的阈值,即第二阈值。若第一数量超过第一阈值,则说明文件块本身包含较多的文件;若第二数量较多,则说明当前批次命中了该目标文件块中的大部分文件;因此,在第一数量超过第一阈值、且第二数量超过第二阈值的情况下,此时由于采用全局字典进行寻址比较复杂,需要多次遍历整个全局字典,不宜构建全局字典,故可以采用批量寻址的方式,即执行上述步骤D11:以批量寻址的方式,批量生成多个目标文件的寻址信息对象。

[0184] 相反地,若第一数量未超过第一阈值、且第二数量未超过第二阈值,则即使创建全局字典,也不会导致全局字典过大,故可以基于全局字典进行寻址,即可以执行上述步骤D12:根据构建的全局字典进行寻址,确定多个目标文件的寻址信息对象。

[0185] 可选地,上述步骤D11“以批量寻址的方式,批量生成多个目标文件的寻址信息对象”可以包括步骤D111至步骤D115。

[0186] 步骤D111,确定当前批次包含的文件标识命中的目标文件块,创建目标文件块的块对象。

[0187] 步骤D112,遍历目标文件块中的文件的头信息,并判断文件的头信息与当前批次包含的文件标识是否相匹配。

[0188] 步骤D113,在文件的头信息与当前批次包含的文件标识相匹配时,根据文件的头信息确定文件在目标文件块中的位置信息;位置信息包括文件的句柄、在目标文件块中的位置和偏移量。

[0189] 步骤D114,根据位置信息生成相匹配文件的寻址信息对象,并将寻址信息对象返回至块对象。

[0190] 步骤D115,在块对象中的寻址信息对象的数量与当前批次包含的文件标识中属于目标文件块的文件标识数量一致时,停止遍历目标文件块。

[0191] 本实施例中,对于当前批次包含的文件标识命中的目标文件块,可以创建该目标文件块的对象(object),即块对象。在进行批量寻址时,遍历目标文件块中的文件,其中,具体是基于文件的头信息实现的遍历,或者说,所遍历的是每个文件的头信息,并不需要遍历文件本身。

[0192] 其中,文件的头信息可以直接或间接地表示文件表示,故基于文件的头信息可以判断其与当前批次包含的文件标识是否相匹配,相匹配的文件即为当前批次需要读取的目标文件,相匹配的文件的头信息即为目标文件的头信息。

[0193] 并且,文件的头信息可以表示文件在目标文件块中的位置信息。具体地,该位置信息可以包括该文件在目标文件块中的句柄、以及位置、偏移量等,该位置信息可以用于形成相应的寻址信息,故基于该位置信息可以生成相匹配文件的寻址信息对象,之后即可将寻址信息对象返回至块对象。

[0194] 若当前批次包含的文件标识均已被匹配,则说明已完成批量寻址,故不再需要遍历目标文件块。具体地,当前批次包含的文件标识中,属于目标文件块的文件标识数量是一定的,设该文件标识数量为a,若通过遍历目标文件块中的文件头信息,当前已查询到a个相匹配的文件头信息,此时,该目标文件块的块对象中也存在a个寻址信息对象;此时,说明当前批次包含的与该目标文件块相关的文件标识均已被匹配,该目标文件块已寻址完毕,故不需要再遍历该目标文件块,可以遍历其他的目标文件块。

[0195] 例如,处理目标文件块的线程,可以判断该目标文件块的文件总数量,即第一数量,以及命中该目标文件块的目标文件的数量,即第二数量;如果命中的目标文件数量较多,例如命中的目标文件数量超过目标文件块聚合文件总量的配置阈值(如1/3),则使用批量寻址的方式。

[0196] 例如,分布式文件系统为CacheFs系统,该目标文件块的块对象设为CacheFsChunkIO,文件的寻址信息对象设为CacheFsChunkFile。在批量寻址的过程中,在创建CacheFsChunkIO对象时,可以将目标文件传参;CacheFsChunkIO对象初始化时,按顺序遍历目标文件块的头文件,判断每次寻址的头信息的小文件名,是否对应目标文件列表中的目标文件,如果命中,则生成该目标文件的寻址信息对象,即CacheFsChunkFile对象,并与小文件名映射,放入到寻址信息对象列表中,待寻址信息对象列表与目标文件列表长度一致,则表示目标文件列表寻址完成,此时可以返回CacheFsChunkFile对象。可以理解,此时一般不需要维护全局的CacheFsChunkFile对象,即不需要维护目标文件块中所有文件的寻址信息对象。

[0197] 可选地,上述步骤D12“根据构建的全局字典进行寻址,确定多个目标文件的寻址信息对象”可以包括步骤D121至步骤D126。

[0198] 步骤D121,判断当前批次包含的文件标识命中的目标文件块是否为第一次命中。

[0199] 步骤D122,在第一次命中目标文件块的情况下,创建目标文件块的块对象。

[0200] 步骤D123,遍历目标文件块中所有文件的头信息,并根据文件的头信息确定文件在目标文件块中的位置信息;位置信息包括文件的句柄、在目标文件块中的位置和偏移量。

[0201] 步骤D124,根据位置信息确定所有文件的寻址信息对象,并将所有文件的寻址信息对象返回至块对象。

[0202] 步骤D125,将块对象放入至全局字典,并从全局字典中读取当前批次包含的文件标识所对应的目标文件的寻址信息对象。

[0203] 步骤D126,在非第一次命中目标文件块的情况下,从全局字典中读取当前批次包含的文件标识所对应的目标文件的寻址信息对象。

[0204] 本实施例中,若目标文件块是第一次命中,则在创建该目标文件块的块对象时,需要确定该目标文件块中所有文件的寻址信息对象,从而形成包含所有寻址信息对象的全局字典。进而,可以在当前批次以及之后的其他批次,基于该全局字典确定目标文件的寻址信息对象,不再需要多次读取文件块,可以减少IO次数。

[0205] 例如,处理目标文件块的线程,可以判断该目标文件块的文件总数量,即第一数量,以及命中该目标文件块的目标文件的数量,即第二数量;如果命中的目标文件数量较少,且目标文件块只包含偏少的文件,则可以通过构建的全局字典进行寻址。

[0206] 例如,分布式文件系统为CacheFs系统,该目标文件块的块对象设为CacheFsChunkIO,文件的寻址信息对象设为CacheFsChunkFile。若当前需要构建全局字典,则在第一次命中目标文件块时,可以直接创建目标文件块的块对象,即CacheFsChunkIO对象,并在初始化该对象时,按顺序遍历该目标文件块的小文件头信息,获取所有小文件在该目标文件块中的位置信息,例如句柄fh、位置pos、偏移量offset等信息,并基于该位置信息生成每个小文件的寻址信息对象,即CacheFsChunkFile对象,并与小文件名映射,放入到CacheFsChunkIO对象的全局字典members(也可称为全局成员字典)中。遍历完成后,将

CacheFsChunkIO对象与目标文件块的文件块名进行映射,并放入到数据集的全局字典chunk_stream中。在Dataloader多进程并发时,每个文件块的全局寻址只遍历一次。之后其它进程命中时,直接从全局字典chunk_stream提取CacheFsChunkIO对象,并找到全局字典members的小文件地址信息,即寻址信息对象。可以理解,该过程只涉及到目标文件块的头文件,以及逻辑数据等处理,当前并未触发对目标文件块中具体文件数据的读取。

[0207] 例如,文件标识为索引,在迭代批量读取数据的过程中,若在当前迭代,训练库接收到的索引列表为[2, 3, 35, 36, 98, 35, 17, 9, 8, 20],可以将索引列表按照索引对应文件所属的文件块进行分组,生成属于同一文件块的文件列表,即目标文件列表,并与相应文件块的文件块名(chunk name)进行映射。

[0208] 例如,其中的[2, 98, 35, 17, 9, 8, 20]为一组,其均属于文件块imagenet_4M_1,且对应的文件具体如下:

```
[0209]  [
[0210]  'imagenet_4M/imagenet_3/imagenet_2/ILSVRC2012_test_00000008.JPEG
[0211]  'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000035.JPEG'
[0212]  'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000006.JPEG'
[0213]  'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000007.JPEG'
[0214]  'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000008.JPEG'
[0215]  'imagenet_4M/imagenet_2/imagenet_1/ILSVRC2012_test_00000014.JPEG'
[0216]  'imagenet_4M/imagenet_2/imagenet_1/ILSVRC2012_test_00000017.JPEG'
[0217]  ]
```

[0218] 其中的[3, 35, 36] 为另一组,其均属于文件块imagenet_4M_2,且对应的文件具体如下:

```
[0219]  [
[0220]  imagenet_4M/imagenet_1/imagenet_1/ILSVRC2012_test_00000015.JPEG
[0221]  imagenet_4M/imagenet_1/imagenet_1/ILSVRC2012_test_00000016.JPEG
[0222]  imagenet_4M/imagenet_1/imagenet_1/ILSVRC2012_test_00000017.JPEG
[0223]  ]
```

[0224] 之后,可以构建包含目标文件块的块对象的全局字典chunk_stream。例如,若全局字典chunk_stream包含上述的文件块imagenet_4M_1的块对象,该全局字典chunk_stream的一种形式可以如下:

```
[0225]  CachefsDataset self.chunk_stream
[0226]  {
[0227]  /mnt/jfs/pack/imagenet_4M_1:<cachefs.CacheFsChunkIO.CacheFsChunkIO
object at 0x7f2db285be10>
[0228]  }
```

[0229] 并且,可以遍历确定目标文件块imagenet_4M_1中每个文件的寻址信息对象CachFsChunkFile,其包括句柄fh、位置pos、偏移量offset等。进而,将所有文件的寻址信息对象返回至块对象,可以形成相应的全局字典members。例如,该全局字典members的一种形式可以如下:

```
[0230] CacheFsChunkIO self.members
[0231] {
[0232] 'imagenet_4M/imagenet_3/imagenet_2/ILSVRC2012_test_00000008.JPEG':<c
achefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4f87e65d90>,
[0233] 'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000035.JPEG':<c
achefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4faf65ee50>,
[0234] 'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000006.JPEG':<ca
chefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4faf65ee58>,
[0235] 'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000007.JPEG':<ca
chefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4faf65ee72>,
[0236] 'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000008.JPEG':<ca
chefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4faf65ee69>,
[0237] 'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000038.JPEG':<c
achefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4faf65ae50>,
[0238] 'imagenet_4M/imagenet_1/imagenet_2/ILSVRC2012_test_00000031.JPEG':<c
achefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4faf65gy50>,
[0239] 'imagenet_4M/imagenet_2/imagenet_1/ILSVRC2012_test_00000013.JPEG':<c
achefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4faf65fg90>,
[0240] 'imagenet_4M/imagenet_2/imagenet_1/ILSVRC2012_test_00000014.JPEG':<c
achefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4faf65hjk0>,
[0241] 'imagenet_4M/imagenet_2/imagenet_1/ILSVRC2012_test_00000017.JPEG':<c
achefs.CacheFsChunkFile.CacheFsChunkFile object at 0x7f4faf65vb35>
[0242] .....
[0243] }
```

[0244] 其中,训练库基于寻址信息对象实现寻址的时序图可参见图14所示。

[0245] 步骤S1105,根据寻址信息从命中的文件块中读取相应的文件。

[0246] 本实施例中,例如,可以根据寻址信息对象快速读取出相应的目标文件。可以理解,若构建有全局字典,则从全局字典中获取寻址信息对象。

[0247] 在一些可选的实施方式中,上述步骤S1105“根据寻址信息从命中的文件块中读取相应的文件”可以包括以下步骤E1至步骤E3。

[0248] 步骤E1,判断当前批次包含的文件标识命中的目标文件块是否为第一次命中。

[0249] 步骤E2,在第一次命中目标文件块的情况下,对目标文件块进行缓存,并根据寻址信息从缓存的目标文件块中读取相应的目标文件。

[0250] 步骤E3,在非第一次命中目标文件块的情况下,根据寻址信息从缓存的目标文件块中读取相应的目标文件。

[0251] 本实施例中,由于每一批次一般只命中少量的文件块,且在之后的批次可能对同一文件块中的其他文件进行读取,故在第一次命中目标文件块时,在读取文件时,对整个目标文件块进行读取,即读取该目标文件块中的所有数据,并进行缓存,例如缓存至本地计算节点的内存或磁盘空间中。在需要读取该目标文件块中的文件时,直接从缓存中读取相应

的目标文件即可,即只需要与存储系统进行一次I/O交互,可以有效降低I/O交互次数。其中,该存储系统例如可以是图2所示的聚合存储的存储空间,也可以是其他第三方的存储系统。例如,该存储系统可以为Ceph、GFS、Lustre、HDFS、Swift等存储系统,本实施例对此不作限定。

[0252] 具体地,如果文件块中任意一个文件被触发读取,则与存储系统交互一次,触发缓存该文件块到内存中或本地缓存磁盘中。在该计算节点,如果本批次及后续批次中,该文件块的其它文件被读取时,则使用寻址策略和寻址信息,直接从已缓存的文件块中读取文件数据,不再需要与存储系统交互,从而提高了文件读取效率。

[0253] 可选地,上述步骤E3“根据寻址信息从缓存的目标文件块中读取相应的目标文件”可以包括以下步骤E31至步骤E32。

[0254] 步骤E31,在目标文件为图片数据的情况下,将目标文件的寻址信息对象传给图像读取函数,以生成图片文件对象;根据图片文件对象读取相应的图片数据,并进行格式转换。

[0255] 步骤E32,在目标文件为文本数据的情况下,根据目标文件的寻址信息对象读取相应的文本数据,并进行格式转换。

[0256] 本实施例中,对于各个文件块的工作线程,在寻址完成后,可以遍历目标文件列表,并根据文件名命中全局字典中的寻址信息对象CacheFsChunkFile。其中,若该目标文件为图片数据,则可以将该寻址信息对象传给图像读取函数,例如Image.open(),从而可以生成相应的图片文件对象;例如,一种图片文件对象可以是:<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x495 at 0x7F5394EE6B50>。之后,在进行转换(transform)时,可以基于该图片文件对象读取相应的图片数据,并进行格式转换。

[0257] 例如,转换后的一种图片数据为:

[0258] tensor([[[0.4851, 0.5536, 0.5878, ..., -1.1589, -1.1932, -1.3130],

[0259] [0.5022, 0.5022, 0.5022, ..., -0.7479, -0.7479, -0.7479],

[0260] [0.5878, 0.6049, 0.5707, ..., -0.6452, -0.6794, -0.6452],

[0261]]

[0262] 可以理解,在转换时,才触发对文件的读取。

[0263] 若当前需要读取文本数据,即目标文件为文本数据,此时可以直接使用寻址信息对象CacheFsChunkFile的read()方法,直接读取文本数据,然后再根据需要转换为指定格式,完成对文本数据的读取。

[0264] 可选地,若维护有全局字典,还需要适时删除其中的数据。具体地,该方法还包括:在全局字典中的寻址信息对象对应的目标文件被读取之后,删除被读取的目标文件所对应的寻址信息对象;在块对象中的任意寻址信息对象对应的目标文件均被读取之后,删除相应的块对象。

[0265] 本实施例中,在文件数据读取完成后,如果生成了全局字典chunk_stream,则需要将该文件所属文件块的CacheFsChunkIO对象的全局字典members中的寻址信息对象删除,如果全局字典members成员信息已全部被删除,则清除全局字典chunk_stream中的该文件块的CacheFsChunkIO对象,减少海量数据时,全局字典的维护量,提高处理效率和命中效率。

[0266] 将基于本实施例提供的读取方法,与直接从NVME/SSD/HDD磁盘读取数据进行对

比,二者的性能对比可参见图15至图17所示。图15、图16、图17分别为基于SSD、NVME、HDD进行性能对比的测试结果图,每个测试结果图均示出了三种不同数据集在第一次迭代(epoch1)时的运行时长(Elapsed Time),该运行时长即为读取耗时,单位为秒(s)。其中,测试条件为批次大小=128,进程数量为16;并对不同大小的数据集进行测试,例如,“4KB-100W”的数据集表示:该数据集包含100万(其中的W表示万)个4KB大小的文件,其他数据集大小“4KB-1000W”、“128KB-100W”等与此相似。测试结果图中的纵坐标表示运行时长。基于该测试结果可知,本实施例提供的读取方法(基于CacheFS分布式系统实现)可以不同幅度地降低运行时长,这进一步证实了,本实施例提供的方法能够加速了AI训练读取海量小文件数据集的过程。

[0267] 在本实施例中还提供了一种数据读取装置,该装置用于实现上述实施例及优选实施方式,已经进行过说明的不再赘述。如以下所使用的,术语“模块”可以是实现预定功能的软件和/或硬件的组合。尽管以下实施例所描述的装置较佳地以软件来实现,但是硬件,或者软件和硬件的组合的实现也是可能并被构想的。

[0268] 本实施例提供一种数据读取装置,如图18所示,包括:

[0269] 获取模块1801,用于获取多个文件块的块标识,并确定所述文件块中每个文件对应的文件标识;所述文件块是对文件数据集中的文件进行聚合所生成的数据块;

[0270] 分组模块1802,用于对所述多个文件块的块标识进行分组,形成多个块标识组;

[0271] 乱序模块1803,用于对所述多个块标识组进行组间乱序,并对至少部分所述块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列;

[0272] 读取模块1804,用于按照所述文件标识序列中文件标识的顺序,分批次读取与当前批次包含的文件标识相对应的文件。

[0273] 在一些可选的实施方式中,所述文件标识为相对应的文件在所述多个文件块所包含的所有文件中的索引;

[0274] 所述乱序模块1803对所述多个块标识组进行组间乱序,并对至少部分所述块标识组中的文件标识进行组内乱序,形成乱序后的文件标识序列,包括:构建二维数组;所述二维数组中的第一维元素表示相应的块标识组,所述第一维元素对应的第二维元素表示相应块标识组中的文件标识;对所述二维数组中的多个所述第一维元素进行乱序,并对至少部分所述第一维元素对应的第二维元素进行乱序;按照所述二维数组中元素乱序后的顺序,将乱序后的所述二维数组转换为一维的全局索引列表;所述全局索引列表作为文件标识序列。

[0275] 在一些可选的实施方式中,分批次读取的文件用于模型训练,且乱序模块1803还用于:在模型训练的第一次迭代过程中,以所述块标识组为单位生成包含多个文件的描述数据的文件列表;对所述文件列表中的描述数据进行乱序,并将乱序后的文件列表按照所述块标识组的顺序,形成包括多个文件列表中描述数据的全局文件列表;将文件的描述数据在所述全局文件列表中的位置下标作为相应文件的索引。

[0276] 在一些可选的实施方式中,所述分组模块1802对所述多个文件块的块标识进行分组,形成多个块标识组,包括:根据所述文件块中的文件数量,确定分组大小;所述文件块中的文件数量与所述分组大小之间为负相关关系;按照所述分组大小对所述多个文件块的块标识进行分组,形成多个块标识组。

[0277] 在一些可选的实施方式中,所述文件块中的文件数量为在文件块大小的限制下所聚合的文件数量,所述分组大小为根据所述文件数据集的规模所设置的大小;或者,所述文件块中的文件数量为 2^m ,所述分组大小为 2^n ;m、n均为整数。

[0278] 在一些可选的实施方式中,所述读取模块1804读取与当前批次包含的文件标识相对应的文件,包括:对当前批次包含的文件标识相对应的文件进行寻址,确定相应的寻址信息;根据所述寻址信息从命中的文件块中读取相应的文件。

[0279] 在一些可选的实施方式中,所述读取模块1804对当前批次包含的文件标识相对应的文件进行寻址,确定相应的寻址信息,包括:确定当前批次包含的文件标识命中的目标文件,并生成所述目标文件的寻址信息对象;所述寻址信息对象包括所述目标文件在相应文件块中的位置信息。

[0280] 在一些可选的实施方式中,所述读取模块1804生成所述目标文件的寻址信息对象,包括:以批量寻址的方式,批量生成多个所述目标文件的寻址信息对象;或者,根据构建的全局字典进行寻址,确定多个所述目标文件的寻址信息对象;所述全局字典用于维护所述目标文件所在的文件块的所有文件的寻址信息对象。

[0281] 在一些可选的实施方式中,所述读取模块1804以批量寻址的方式,批量生成多个所述目标文件的寻址信息对象,包括:确定当前批次包含的文件标识命中的目标文件块,创建所述目标文件块的块对象;遍历所述目标文件块中的文件的头信息,并判断所述文件的头信息与所述当前批次包含的文件标识是否相匹配;在所述文件的头信息与所述当前批次包含的文件标识相匹配时,根据所述文件的头信息确定所述文件在所述目标文件块中的位置信息;所述位置信息包括文件的句柄、在所述目标文件块中的位置和偏移量;根据所述位置信息生成相匹配文件的寻址信息对象,并将所述寻址信息对象返回至块对象;在所述块对象中的寻址信息对象的数量与所述当前批次包含的文件标识中属于所述目标文件块的文件标识数量一致时,停止遍历所述目标文件块。

[0282] 在一些可选的实施方式中,所述读取模块1804根据构建的全局字典进行寻址,确定多个所述目标文件的寻址信息对象,包括:判断当前批次包含的文件标识命中的目标文件块是否为第一次命中;在第一次命中所述目标文件块的情况下,创建所述目标文件块的块对象;遍历所述目标文件块中所有文件的头信息,并根据所述文件的头信息确定所述文件在所述目标文件块中的位置信息;所述位置信息包括文件的句柄、在所述目标文件块中的位置和偏移量;根据所述位置信息确定所有文件的寻址信息对象,并将所有文件的所述寻址信息对象返回至块对象;将所述块对象放入至全局字典,并从所述全局字典中读取当前批次包含的文件标识所对应的目标文件的寻址信息对象;在非第一次命中所述目标文件块的情况下,从所述全局字典中读取当前批次包含的文件标识所对应的目标文件的寻址信息对象。

[0283] 在一些可选的实施方式中,该装置还包括删除模块,用于:在所述全局字典中的寻址信息对象对应的目标文件被读取之后,删除被读取的目标文件所对应的寻址信息对象;在所述块对象中的任意寻址信息对象对应的目标文件均被读取之后,删除相应的块对象。

[0284] 在一些可选的实施方式中,所述读取模块1804生成所述目标文件的寻址信息对象,包括:确定当前批次包含的文件标识命中的目标文件块中文件的第一数量,并确定当前批次包含的文件标识命中的、属于所述目标文件块的目标文件的第二数量;在所述第一数

量超过第一阈值、且所述第二数量超过第二阈值的情况下,以批量寻址的方式,批量生成多个所述目标文件的寻址信息对象;在所述第一数量未超过第一阈值、且所述第二数量未超过第二阈值的情况下,根据构建的全局字典进行寻址,确定多个所述目标文件的寻址信息对象。

[0285] 在一些可选的实施方式中,所述读取模块1804对当前批次包含的文件标识相对应的文件进行寻址,确定相应的寻址信息,包括:根据当前批次包含的文件标识确定当前批次命中的目标文件所属的文件块;将属于同一文件块的目标文件归为一组,生成包含至少一个目标文件所对应文件标识的目标文件列表;以所述目标文件列表为单位,分别确定每个所述目标文件列表所对应的目标文件的寻址信息。

[0286] 在一些可选的实施方式中,所述读取模块1804还用于:开启多个工作进程,基于所述工作进程读取相应批次的文件;不同的所述工作进程用于读取不同批次的文件,且在所述工作进程读取完当前批次的文件后,所述工作进程用于读取下一批次的文件,直至读取完所述文件数据集;在当前批次的文件读取过程中,开启与命中的文件块数量相匹配的工作线程,每个所述工作线程分别用于确定相应文件块中文件的寻址信息,并读取相应的文件。

[0287] 在一些可选的实施方式中,所述读取模块1804根据所述寻址信息从命中的文件块中读取相应的文件,包括:判断当前批次包含的文件标识命中的目标文件块是否为第一次命中;在第一次命中所述目标文件块的情况下,对所述目标文件块进行缓存,并根据所述寻址信息从缓存的目标文件块中读取相应的目标文件;在非第一次命中所述目标文件块的情况下,根据所述寻址信息从缓存的目标文件块中读取相应的目标文件。

[0288] 在一些可选的实施方式中,所述读取模块1804根据所述寻址信息从缓存的目标文件块中读取相应的目标文件,包括:

[0289] 在所述目标文件为图片数据的情况下,将所述目标文件的寻址信息对象传给图像读取函数,以生成图片文件对象;根据所述图片文件对象读取相应的图片数据,并进行格式转换;

[0290] 在所述目标文件为文本数据的情况下,根据所述目标文件的寻址信息对象读取出相应的文本数据,并进行格式转换。

[0291] 上述各个模块和单元的功能描述与上述对应实施例相同,在此不再赘述。

[0292] 本实施例中的数据读取装置是以功能单元的形式来呈现,这里的单元是指ASIC(Application Specific Integrated Circuit,专用集成电路)电路,包括执行一个或多个软件或固定程序的处理器和存储器,和/或其他可以提供上述功能的器件。

[0293] 本发明实施例还提供一种计算机设备,可以包括上述图18所示的数据读取装置。

[0294] 请参阅图19,图19是本发明可选实施例提供的一种计算机设备的结构示意图,如图19所示,该计算机设备包括:一个或多个处理器10、存储器20,以及用于连接各部件的接口,包括高速接口和低速接口。各个部件利用不同的总线互相通信连接,并且可以被安装在公共主板上或者根据需要以其它方式安装。处理器可以对在计算机设备内执行的指令进行处理,包括存储在存储器中或者存储器上以在外部输入/输出装置(诸如,耦合至接口的显示设备)上显示GUI的图形信息的指令。在一些可选的实施方式中,若需要,可以将多个处理

器和/或多条总线与多个存储器一起使用。同样,可以连接多个计算机设备,各个设备提供部分必要的操作(例如,作为服务器阵列、一组刀片式服务器、或者多处理器系统)。图19中以一个处理器10为例。

[0295] 处理器10可以是中央处理器,网络处理器或其组合。其中,处理器10还可以进一步包括硬件芯片。上述硬件芯片可以是专用集成电路,可编程逻辑器件或其组合。上述可编程逻辑器件可以是复杂可编程逻辑器件,现场可编程逻辑门阵列,通用阵列逻辑或其任意组合。

[0296] 其中,所述存储器20存储有可由至少一个处理器10执行的指令,以使所述至少一个处理器10执行实现上述实施例示出的方法。

[0297] 存储器20可以包括存储程序区和存储数据区,其中,存储程序区可存储操作系统、至少一个功能所需要的应用程序;存储数据区可存储根据计算机设备的使用所创建的数据等。此外,存储器20可以包括高速随机存取存储器,还可以包括非瞬时存储器,例如至少一个磁盘存储器件、闪存器件、或其他非瞬时固态存储器件。在一些可选的实施方式中,存储器20可选包括相对于处理器10远程设置的存储器,这些远程存储器可以通过网络连接至该计算机设备。上述网络的实例包括但不限于互联网、企业内部网、局域网、移动通信网及其组合。

[0298] 存储器20可以包括易失性存储器,例如,随机存取存储器;存储器也可以包括非易失性存储器,例如,快闪存储器,硬盘或固态硬盘;存储器20还可以包括上述种类的存储器的组合。

[0299] 该计算机设备还包括输入装置30和输出装置40。处理器10、存储器20、输入装置30和输出装置40可以通过总线或者其他方式连接,图19中以通过总线连接为例。

[0300] 输入装置30可接收输入的数字或字符信息,以及产生与该计算机设备的用户设置以及功能控制有关的键信号输入,例如触摸屏、小键盘、鼠标、轨迹板、触摸板、指示杆、一个或者多个鼠标按钮、轨迹球、操纵杆等。输出装置40可以包括显示设备、辅助照明装置(例如,LED)和触觉反馈装置(例如,振动电机)等。上述显示设备包括但不限于液晶显示器,发光二极管,显示器和等离子体显示器。在一些可选的实施方式中,显示设备可以是触摸屏。

[0301] 本发明实施例还提供了一种计算机可读存储介质,上述根据本发明实施例的方法可在硬件、固件中实现,或者被实现为可记录在存储介质,或者被实现通过网络下载的原始存储在远程存储介质或非暂时机器可读存储介质中并将被存储在本地存储介质中的计算机代码,从而在此描述的方法可被存储在使用通用计算机、专用处理器或者可编程或专用硬件的存储介质上的这样的软件处理。其中,存储介质可为磁碟、光盘、只读存储记忆体、随机存储记忆体、快闪存储器、硬盘或固态硬盘等;进一步地,存储介质还可以包括上述种类的存储器的组合。可以理解,计算机、处理器、微处理器控制器或可编程硬件包括可存储或接收软件或计算机代码的存储组件,当软件或计算机代码被计算机、处理器或硬件访问且执行时,实现上述实施例示出的方法。

[0302] 本申请中的方法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。所述计算机程序产品包括一个或多个计算机程序或指令。在计算机上加载和执行所述计算机程序或指令时,全部或部分地执行本申请所述的流程或功能。所述计算机可以是通用计算机、专用计算机、

计算机网络、网络设备、用户设备、核心网设备、OAM或者其它可编程装置。

[0303] 所述计算机程序或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,所述计算机程序或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或无线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。所述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集成一个或多个可用介质的服务器、数据中心等数据存储设备。所述可用介质可以是磁性介质,例如,软盘、硬盘、磁带;也可以是光介质,例如,数字视频光盘;还可以是半导体介质,例如,固态硬盘。该计算机可读存储介质可以是易失性或非易失性存储介质,或可包括易失性和非易失性两种类型的存储介质。

[0304] 虽然结合附图描述了本发明的实施例,但是本领域技术人员可以在不脱离本发明的精神和范围的情况下做出各种修改和变型,这样的修改和变型均落入由所附权利要求所限定的范围之内。

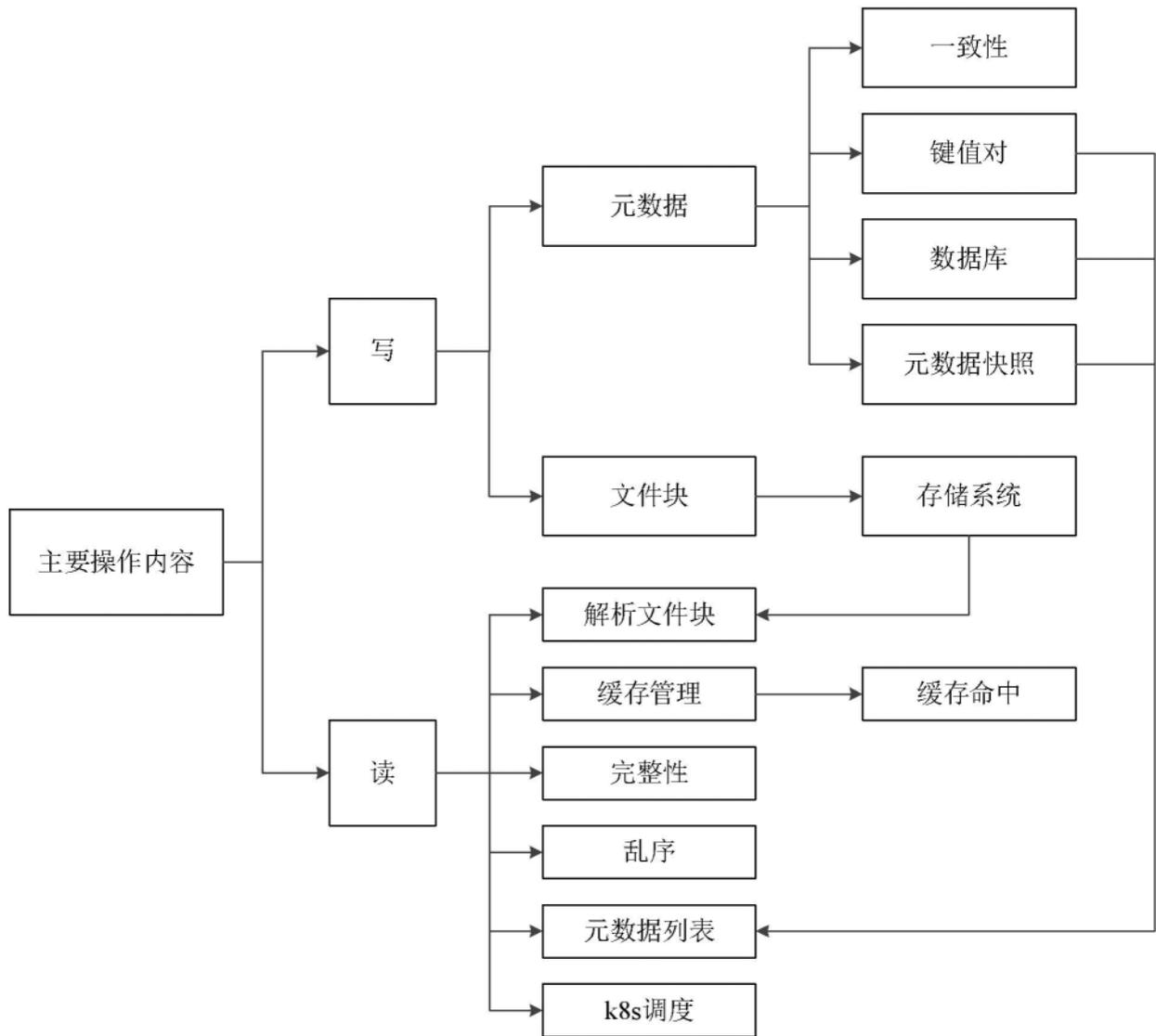


图1

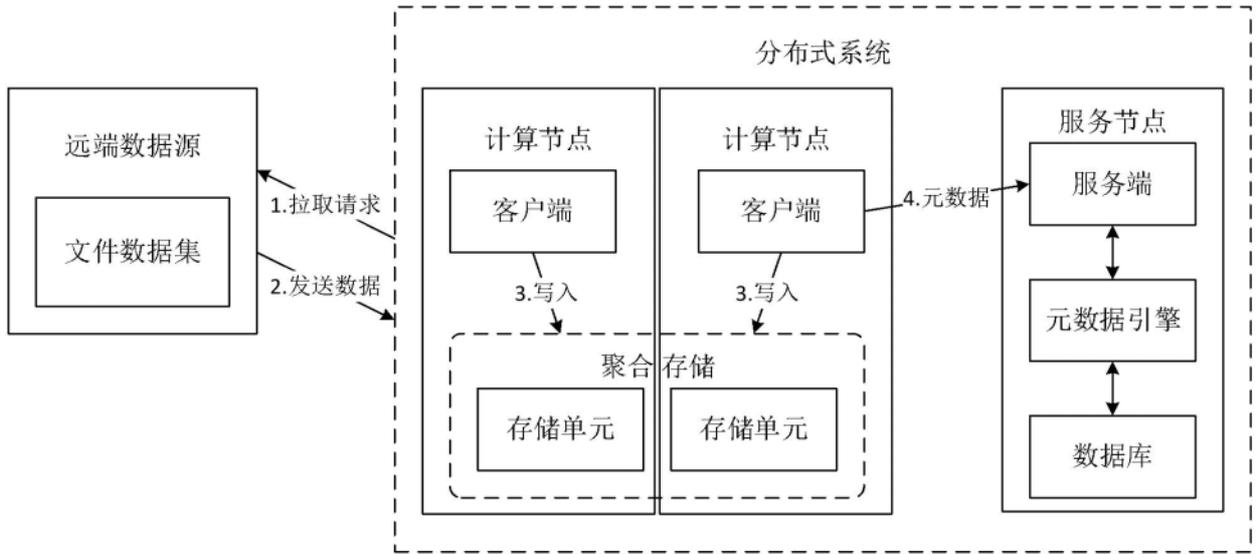


图2

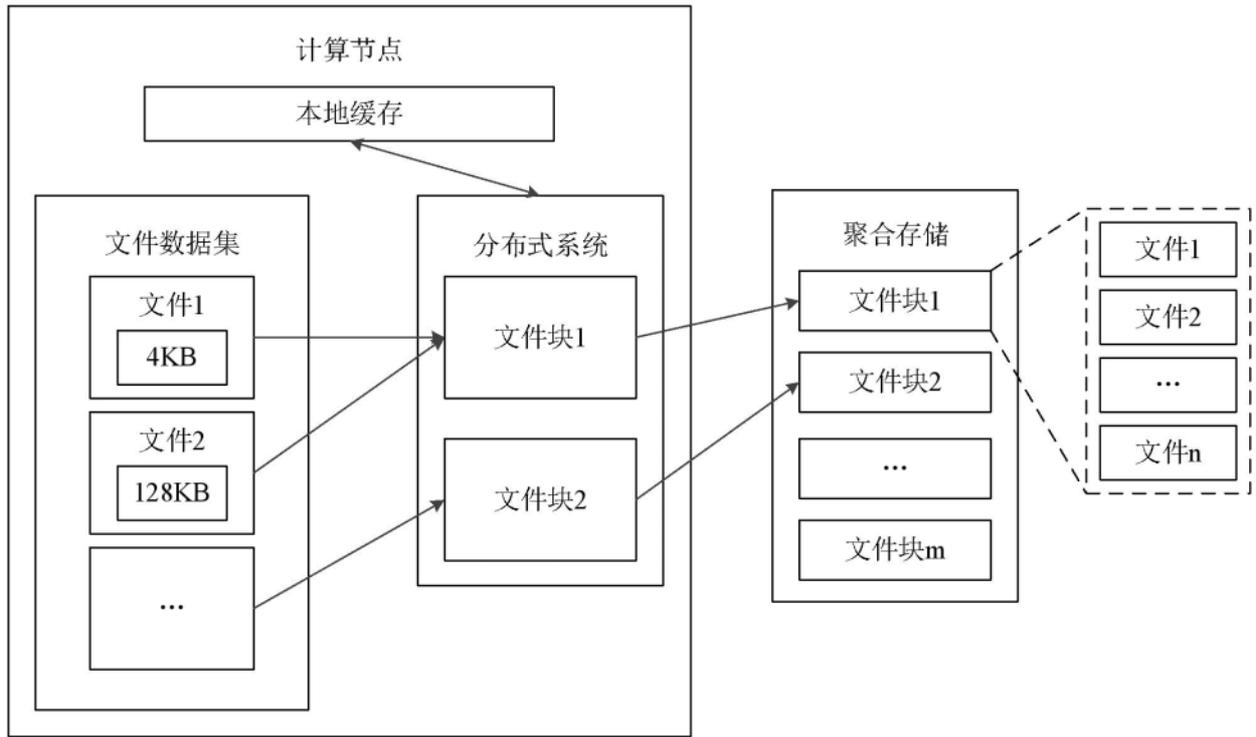


图3

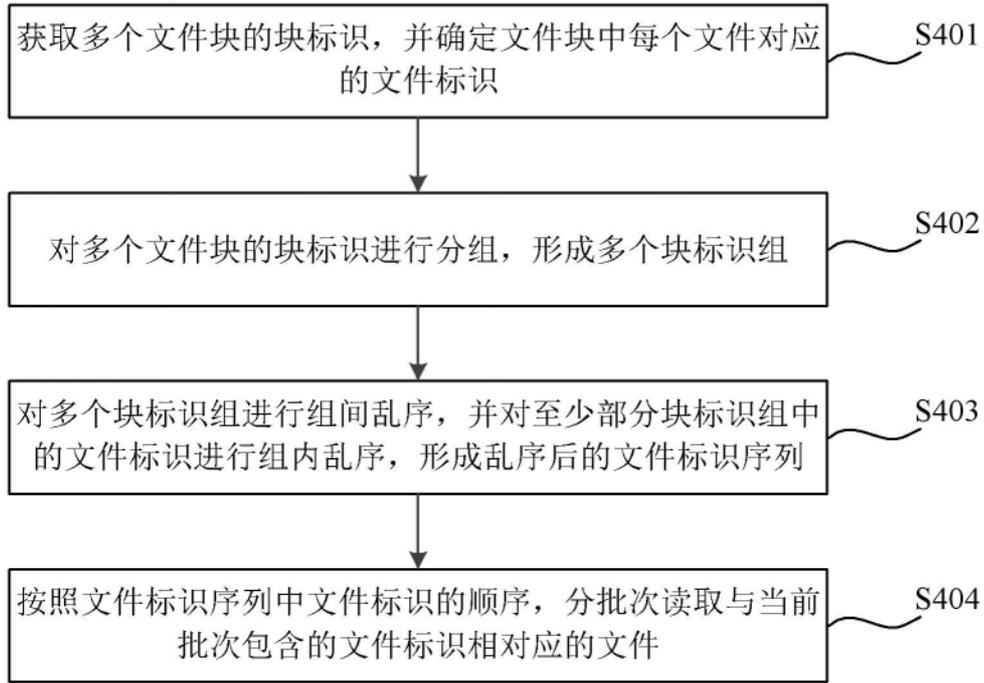


图4

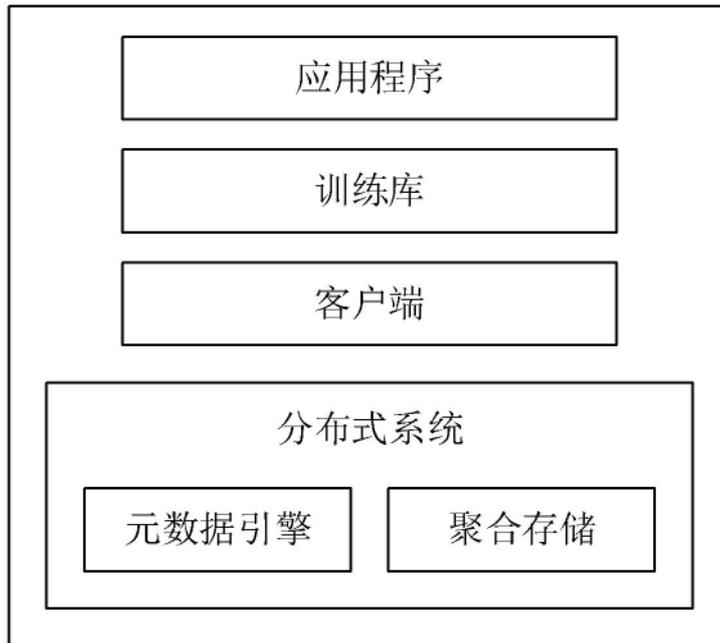


图5

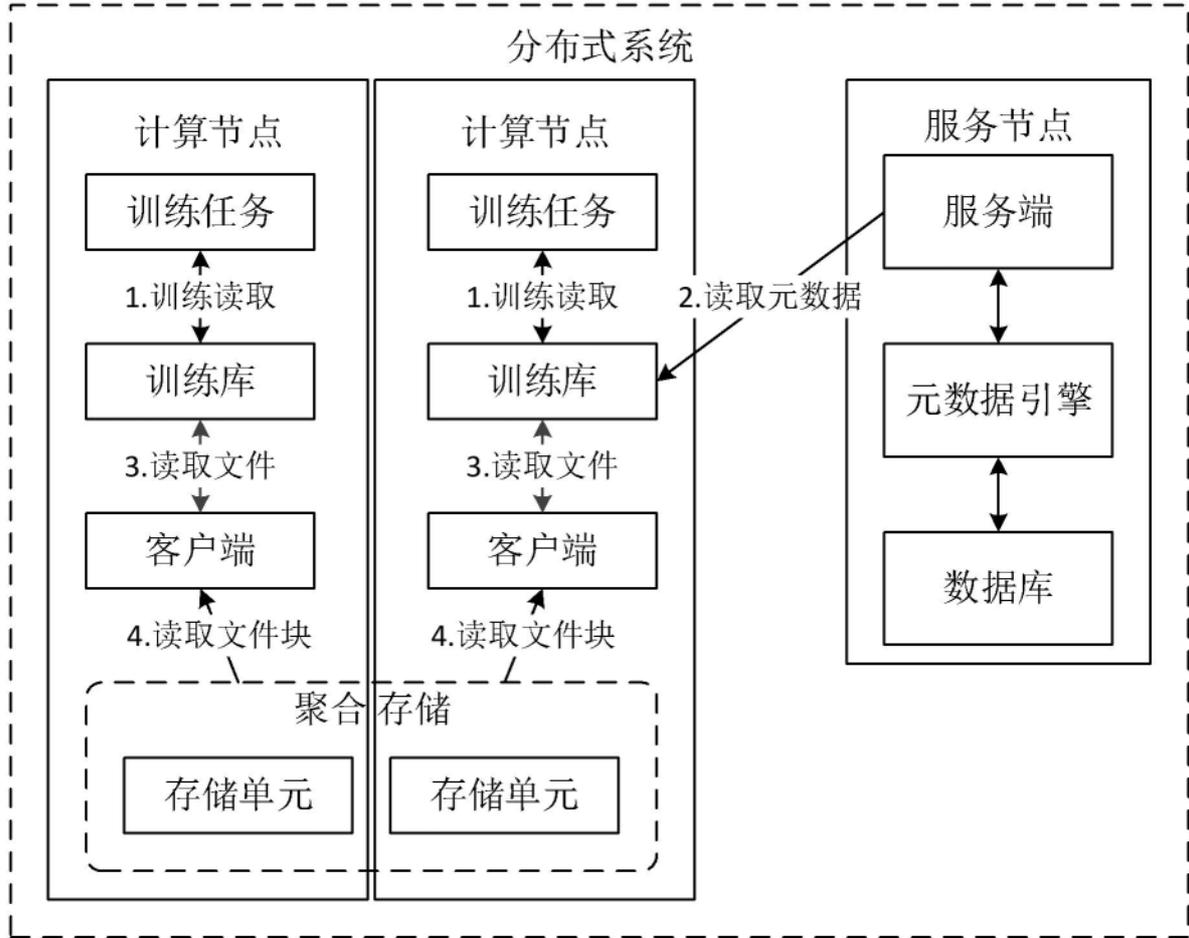


图6

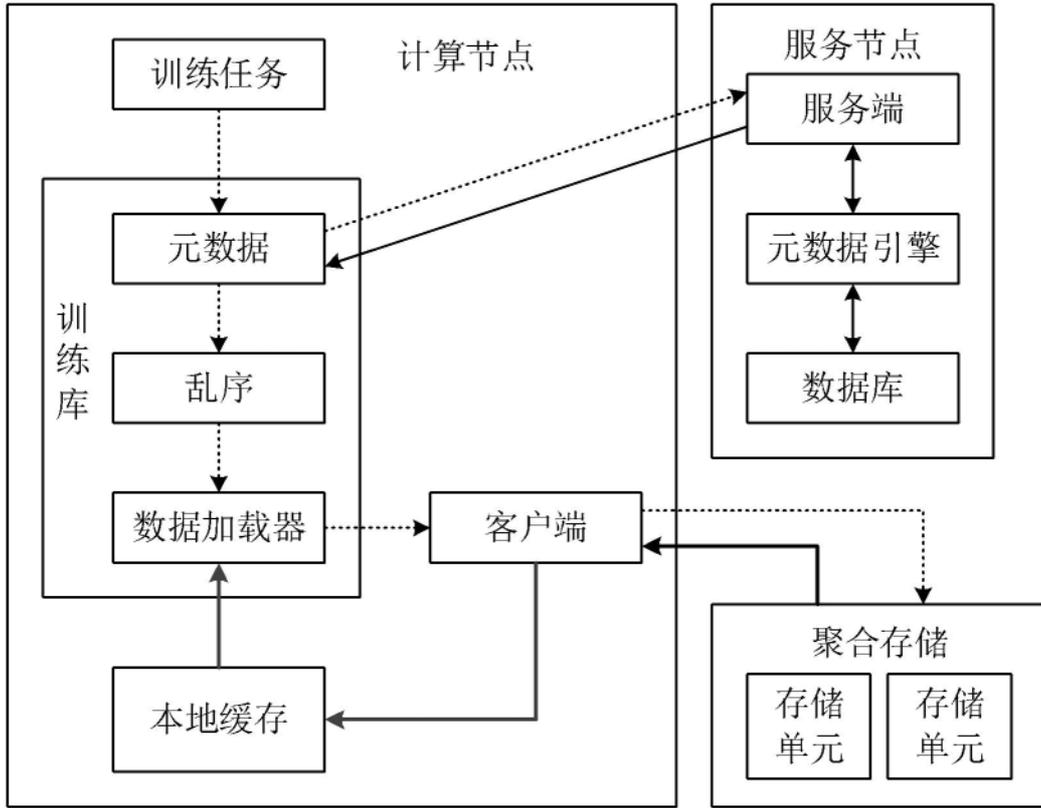


图7

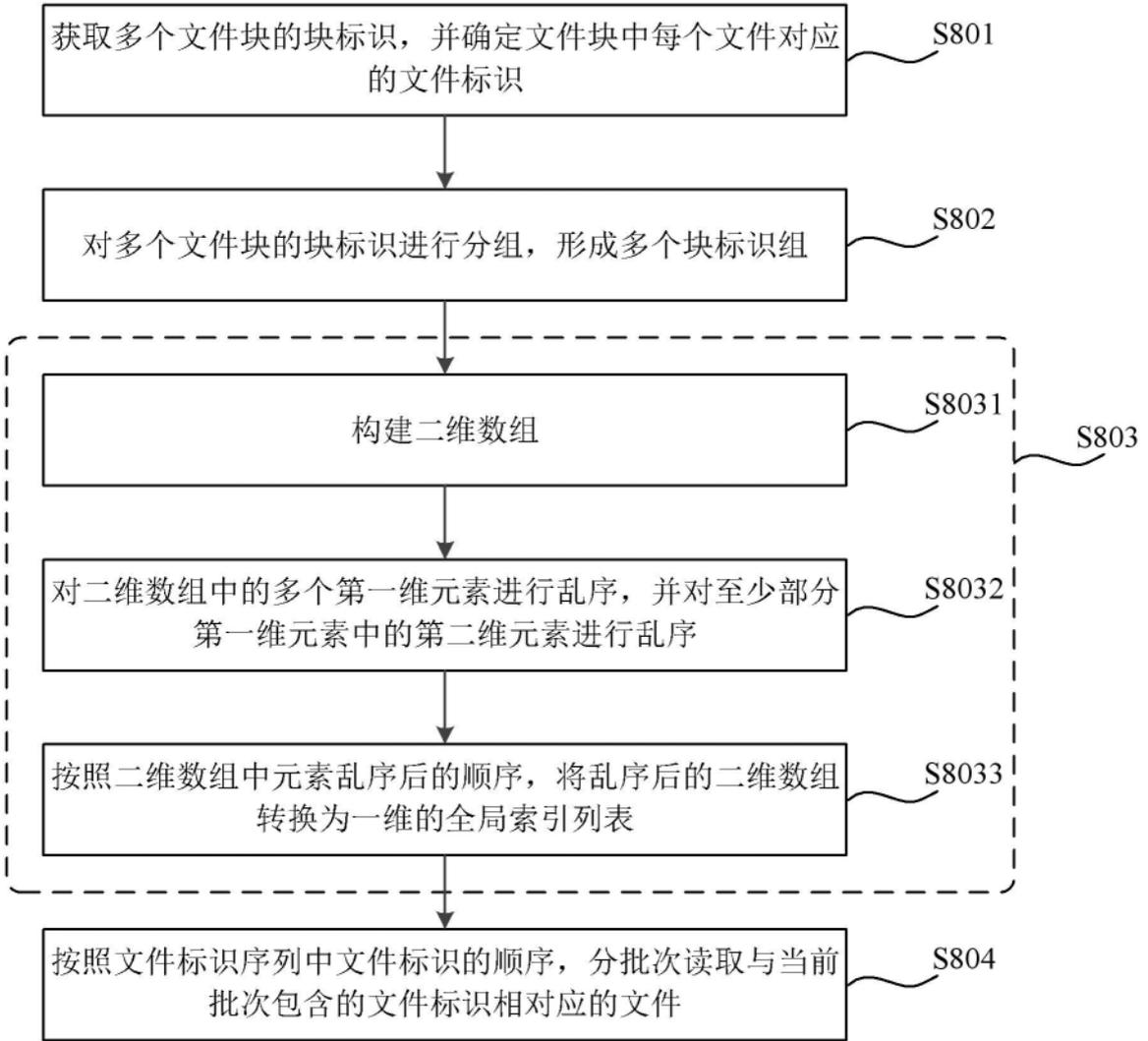


图8

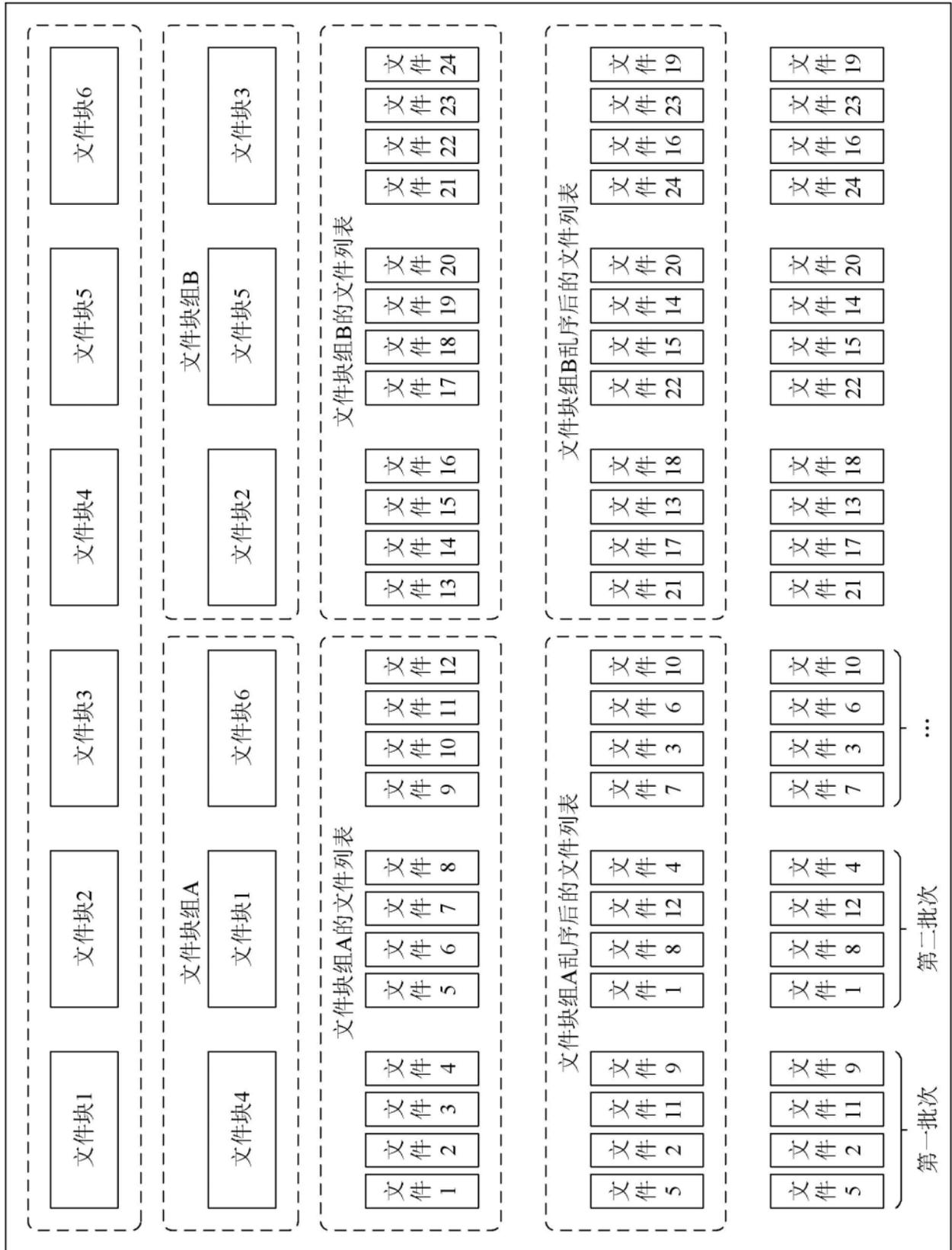


图9

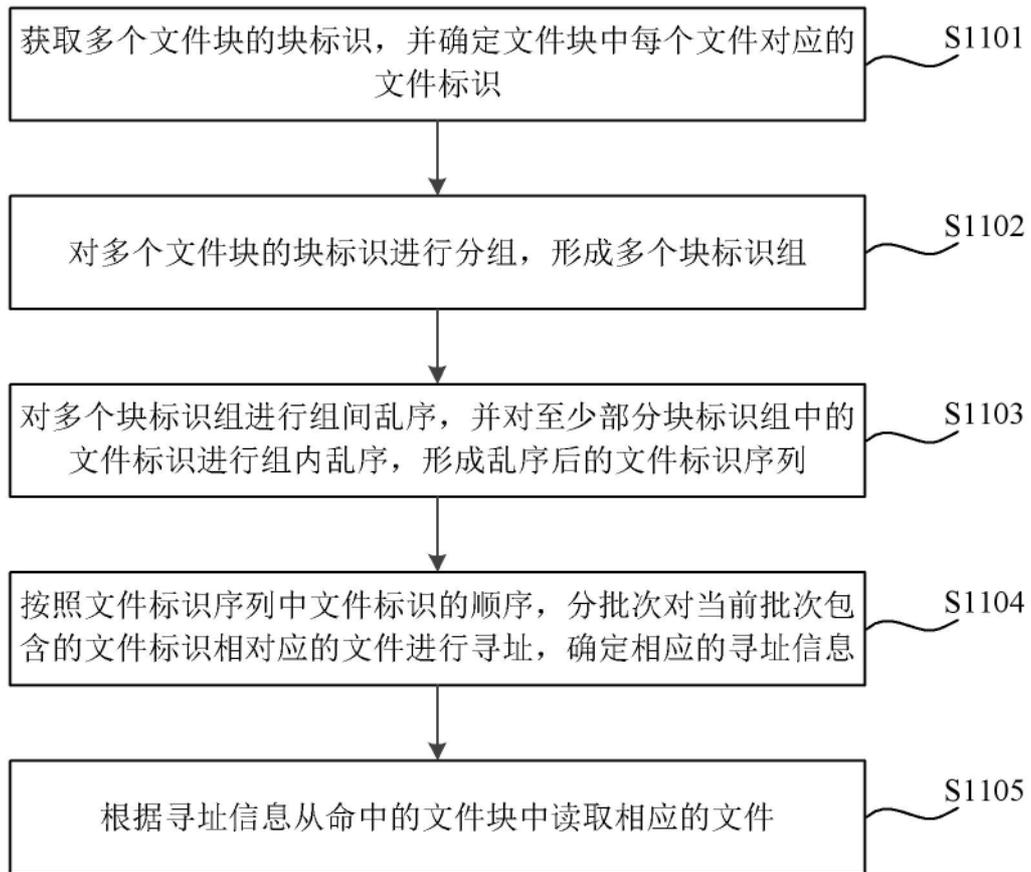


图11

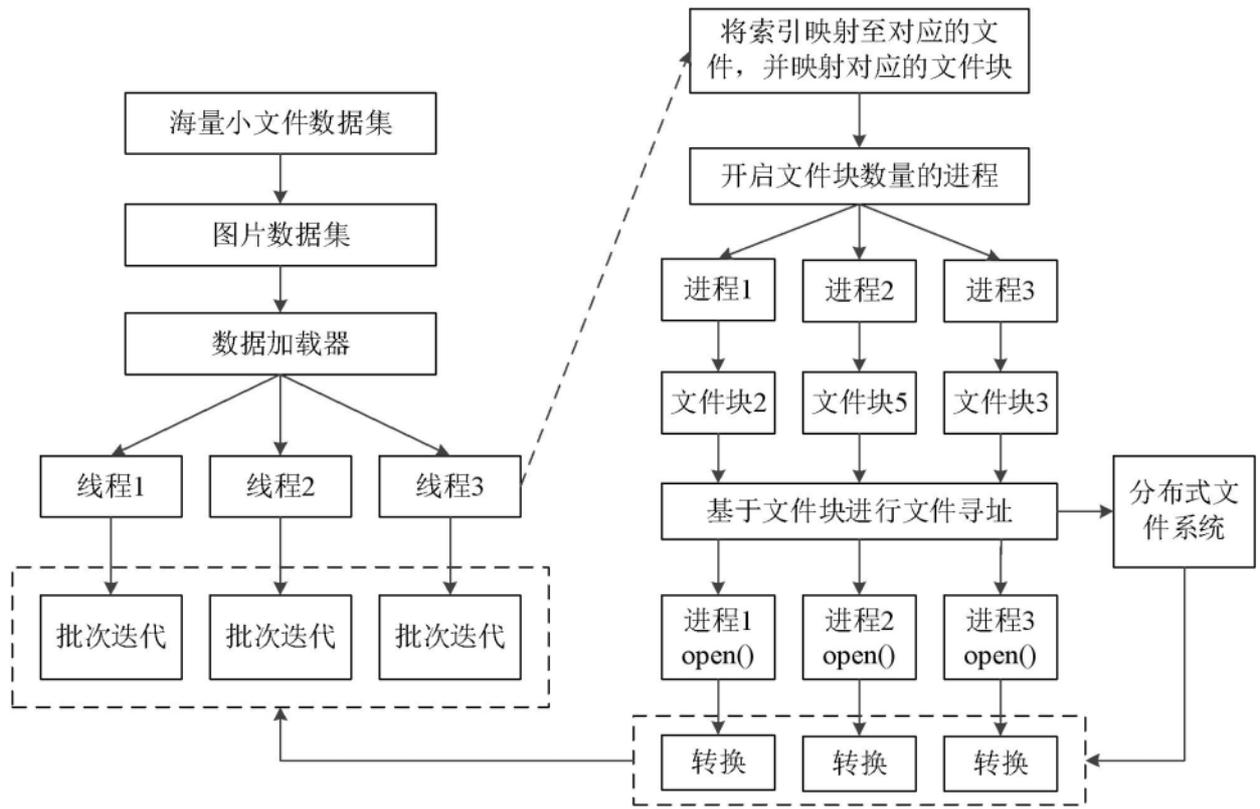


图12

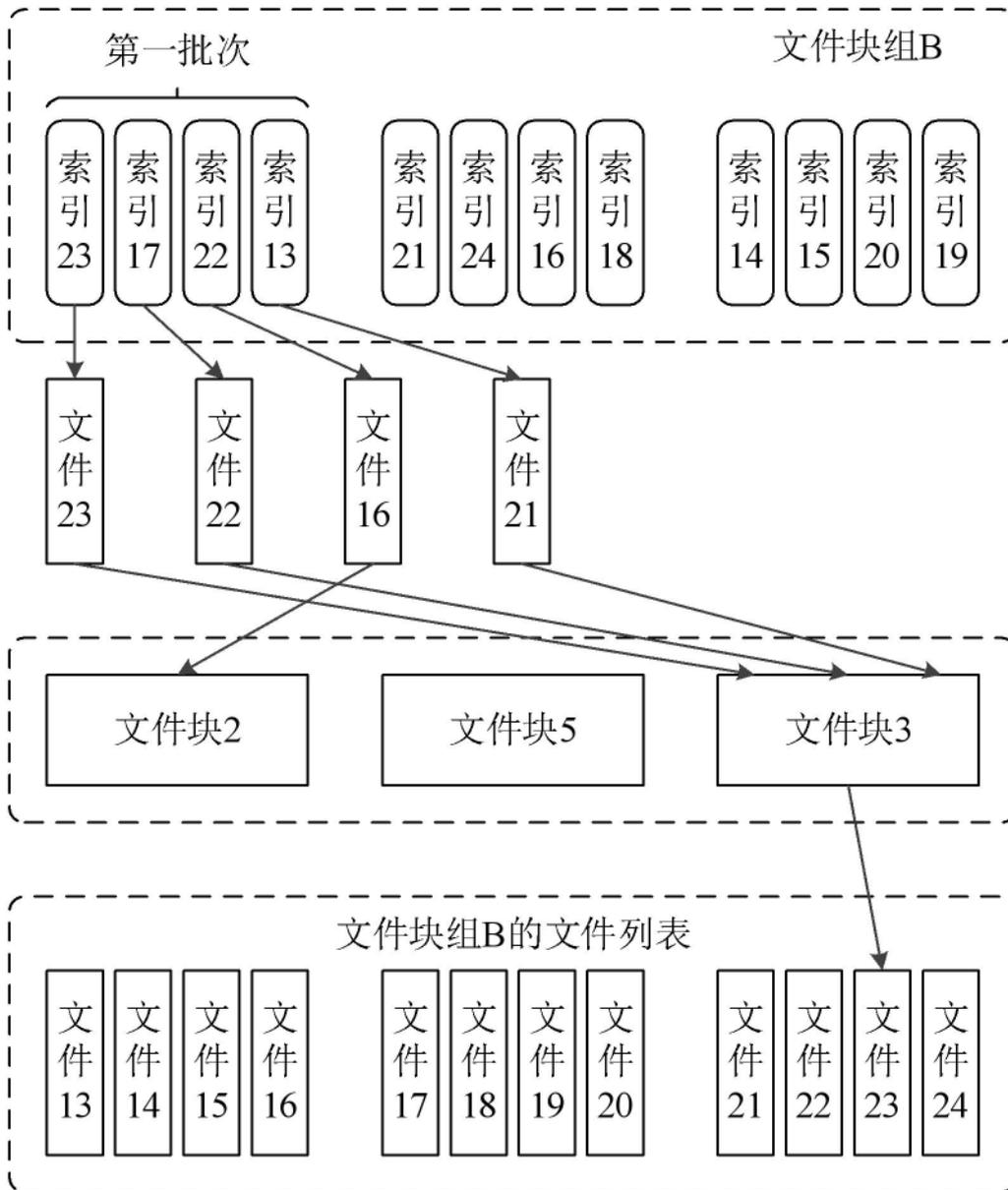


图13

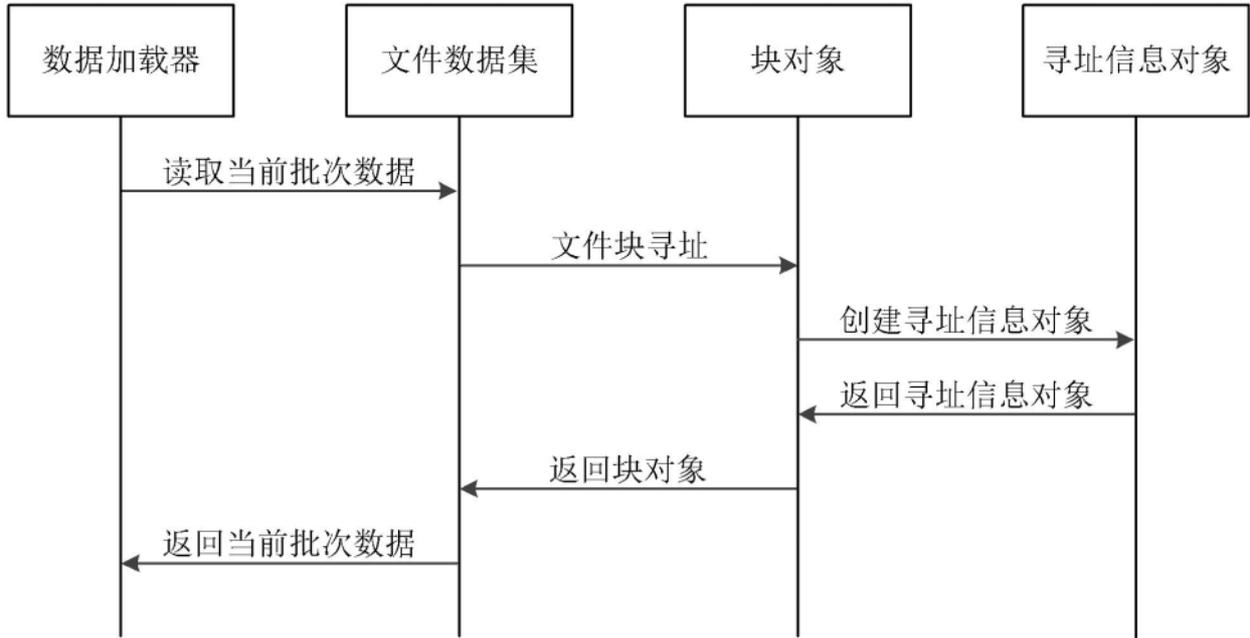


图14

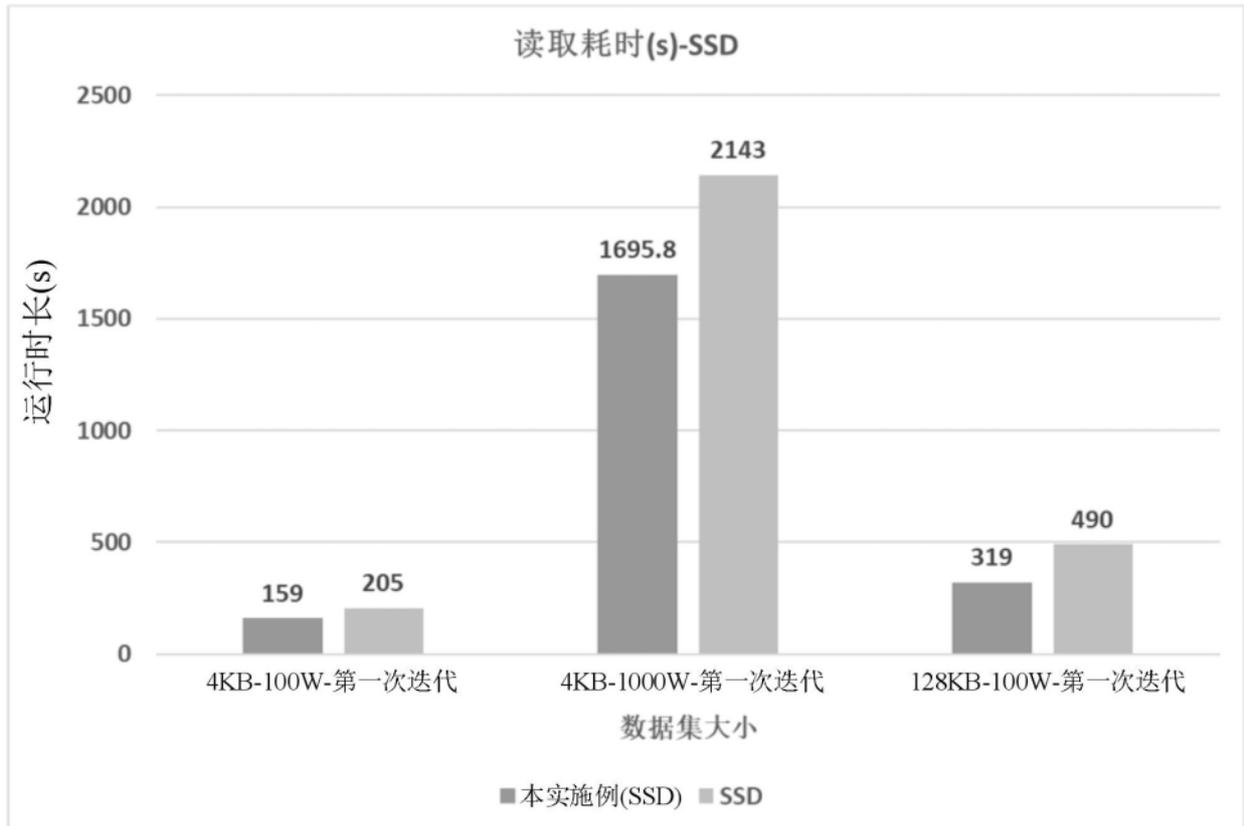


图15

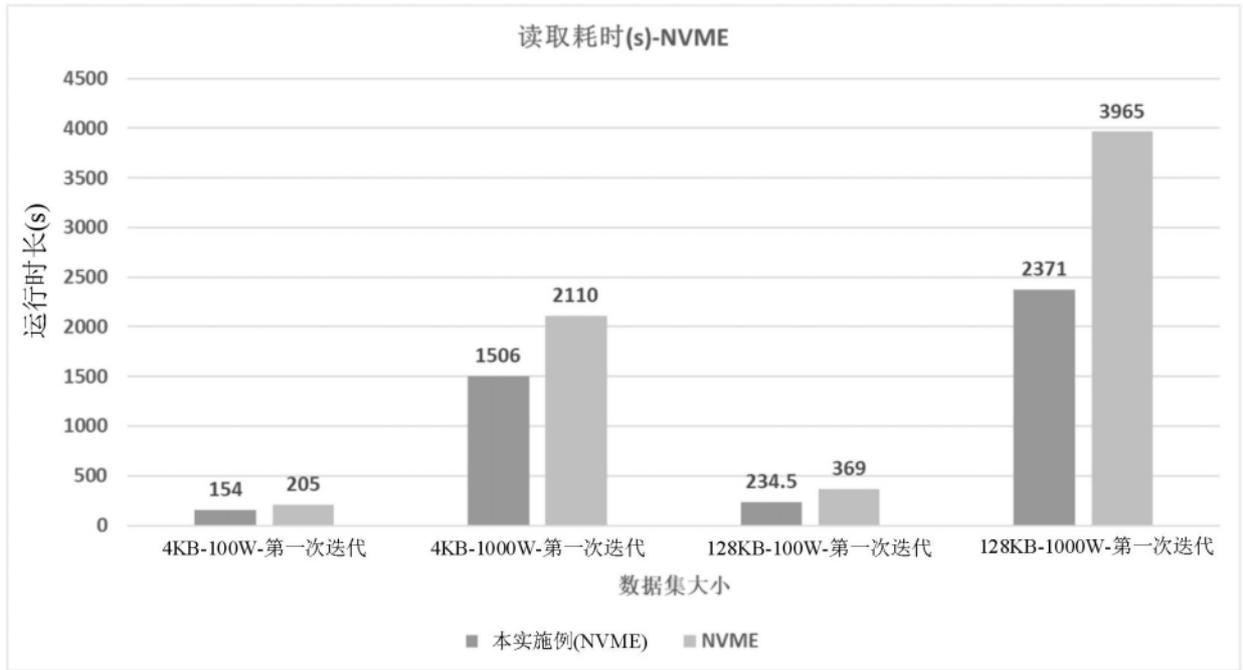


图16

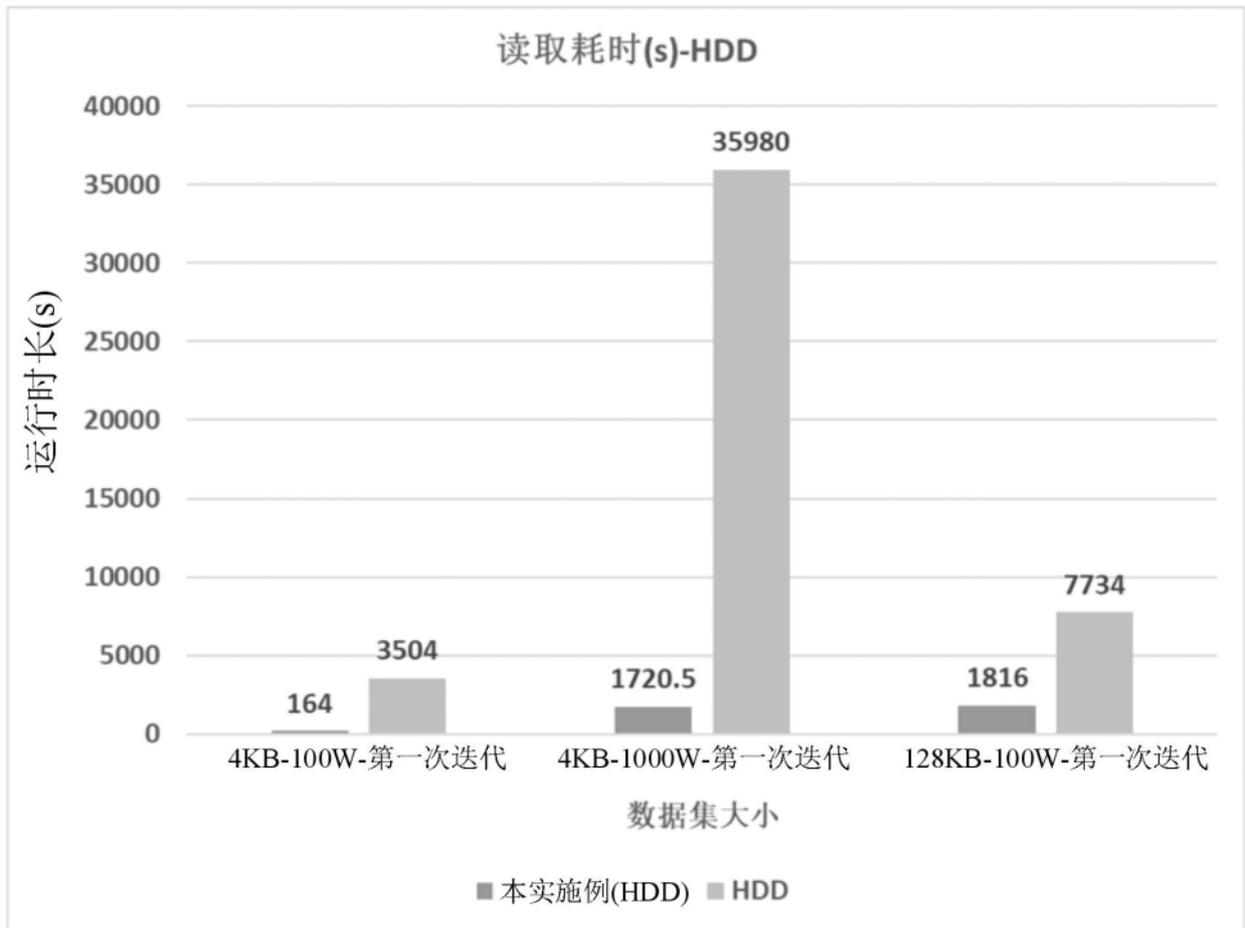


图17

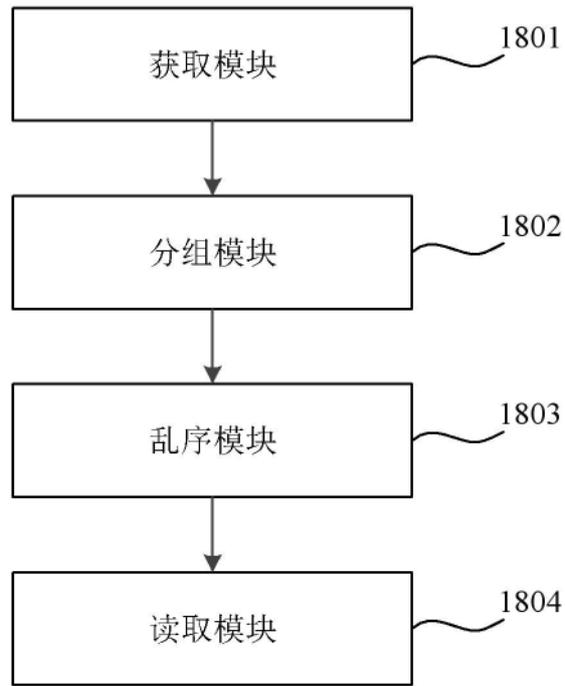


图18

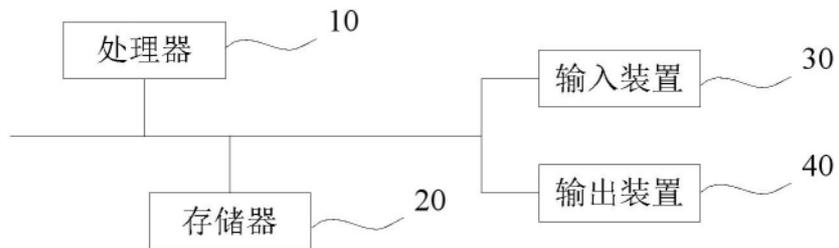


图19