(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 113688096 B
(45)授权公告日 2024.08.23

(21)申请号 202110799890.7

(22)申请日 2021.07.15

(65)同一申请的已公布的文献号
申请公布号 CN 113688096 A

(43)申请公布日 2021.11.23

(73)专利权人 三星(中国)半导体有限公司
地址 710000 陕西省西安市高新区沣河北路1999号
专利权人 三星电子株式会社

(72)发明人 李玉芳 曹文彬 闫浩 薛丽娟 石沙

(74)专利代理机构 北京铭硕知识产权代理有限公司 11286
专利代理师 王艳茹 苏银虹

(51)Int.Cl.
G06F 16/11(2019.01)

(56)对比文件
CN 106682184 A,2017.05.17
CN 109522283 A,2019.03.26

审查员 曹彦

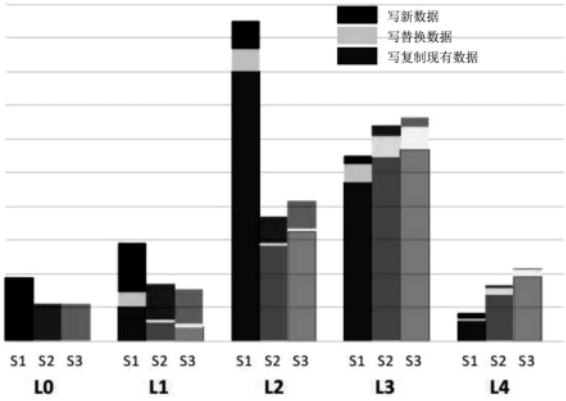
权利要求书3页 说明书11页 附图8页

(54)发明名称

存储方法、存储装置和存储系统

(57)摘要

公开了一种存储方法、存储装置和存储系统,该存储方法包括:确定数据库中将要合并的层;基于数据库中将要合并的层的文件确定将要修改的数据块;根据将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组;在重组后的数据块的字节大小未超过第一预设值时,将存储器中将要修改的数据块替换为重组后的数据块;将重组后的数据块的键和统计信息存入数据库对应的文件。



CN 113688096 B

1.一种存储方法,包括:

确定数据库中将要合并的层;

基于所述将要合并的层的可变长文件确定将要修改的数据块;

根据所述将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组,其中,所述存储器用于存储数据块且存储的数据块的字节大小是可变的;

在重组后的数据块的字节大小未超过第一预设值时,将所述存储器中将要修改的数据块替换为重组后的数据块;

将所述重组后的数据块的键和统计信息存入所述数据库对应的可变长文件;

其中,所述根据所述将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组包括:

根据所述将要修改的数据块对应的用户键在存储器中读取对应的数据块,其中,所述用户键包含在与所述数据块对应的键中;

从所述对应的数据块中筛选出每一个用户键对应的最大序列号对应的数据,其中,所述序列号包含在与所述数据块对应的键中,用于指示同一个用户键出现的顺序;

将筛选出的数据按用户键进行排序后作为重组后的数据块。

2.根据权利要求1所述的存储方法,其特征在于,还包括:

在所述重组后的数据块的字节大小超过所述第一预设值时,按所述第一预设值将重组后的数据块进行拆分;

将所述存储器中将要修改的数据块替换为拆分后的数据块;

所述将所述重组后的数据块的键和统计信息存入所述数据库对应的可变长文件包括:将拆分后的数据块的键和统计信息存入所述数据库对应的可变长文件。

3.根据权利要求2所述的存储方法,其特征在于,在将所述重组后的数据块的键和统计信息存入所述数据库对应的可变长文件之后,还包括:

判断所述对应的可变长文件的所有数据块的字节大小是否超过第二预设值;

在判断结果为是时,按所述第二预设值将所述对应的可变长文件进行拆分,将所述重组后的数据块的键和统计信息以及所述对应的可变长文件的未修改的数据块的键和统计信息,按键的顺序分别存入所述拆分后的可变长文件;

在判断结果为否时,将所述重组后的数据块的所述键和所述统计信息存入所述对应的可变长文件。

4.根据权利要求1所述的存储方法,其特征在于,所述确定数据库中将要合并的层,包括:

检测到所述数据库中的可变长文件增加;

获取所述数据库中每个层的字节大小;

将每个层中字节大小超过对应层的字节阈值的层作为前一层;

将所述前一层和后一层确定为将要合并的层,其中,所述后一层为所述前一层相邻的下一层。

5.根据权利要求4所述的存储方法,其特征在于,所述检测到所述数据库中的可变长文件增加,包括:

在写入的数据的字节大小超过第一预定阈值时,确定所述数据库中的可变长文件增

加。

6.根据权利要求1所述的存储方法,其特征在于,所述基于所述将要合并的层的可变长文件确定将要修改的数据块包括:

获取所述将要合并的层中的前一层中文件字节最大的文件和预定文件,其中,所述预定文件为所述前一层中包含文件字节最大的文件的用户键的文件,所述用户键包含在与所述文件中的数据块对应的键中;

基于所述文件字节最大的文件的用户键和所述预定文件的用户键,得到用户键范围;

获取所述将要合并的层中的后一层中,用户键与所述用户键范围存在重叠的数据块,以及所述将要合并的层中的前一层中,用户键与所述用户键范围存在重叠的数据块;

将重叠的数据块确定为将要修改的数据块。

7.一种存储装置,包括:

第一确定单元,用于确定数据库中将要合并的层;

第二确定单元,用于所述将要合并的层的可变长文件确定将要修改的数据块;

重组单元,用于根据所述将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组,其中,所述存储器用于存储数据块且存储的数据块的字节大小是可变的;

替换单元,用于在重组后的数据块的字节大小未超过第一预设值时,将所述存储器中将要修改的数据块替换为重组后的数据块;

写入单元,用于将所述重组后的数据块的键和统计信息存入所述数据库对应的可变长文件;

其中,所述重组单元,还用于根据所述将要修改的数据块对应的用户键在存储器中读取对应的数据块,其中,所述用户键包含在与所述数据块对应的键中;从所述对应的数据块中筛选出每一个用户键对应的最大序列号对应的数据,其中,所述序列号包含在与所述数据块对应的键中,用于指示同一个用户键出现的顺序;将筛选出的数据按用户键进行排序后作为重组后的数据块。

8.根据权利要求7所述的存储装置,其特征在于,所述替换单元,还用于在重组后的数据块的字节大小超过所述第一预设值时,按所述第一预设值将重组后的数据块进行拆分;将所述存储器中的将要修改的数据块替换为拆分后的数据块;所述写入单元,还用于将拆分后的数据块的键和统计信息存入所述数据库对应的可变长文件。

9.根据权利要求8所述的存储装置,其特征在于,所述写入单元,还用于在将所述重组后的数据块的键和统计信息存入所述数据库对应的可变长文件之后,判断所述对应的可变长文件的所有数据块的字节大小是否超过第二预设值;在判断结果为是时,按所述第二预设值将所述对应的可变长文件进行拆分,将重组后的数据块的键和统计信息以及所述对应的可变长文件的未修改的数据块的键和统计信息,按键的顺序分别存入所述拆分后的可变长文件;在判断结果为否时,将所述重组后的数据块的所述键和所述统计信息存入所述对应的可变长文件。

10.根据权利要求7所述的存储装置,其特征在于,所述第一确定单元,还用于检测到所述数据库中的可变长文件增加;获取所述数据库中每个层的字节大小;将每个层中字节大小超过对应层的字节阈值的层作为前一层;将所述前一层和后一层确定为将要合并的层,

其中,所述后一层为所述前一层相邻的下一层。

11.根据权利要求10所述的存储装置,其特征在于,所述第一确定单元,还用于在写入的数据的字节大小超过第一预定阈值时,确定所述数据库中的可变长文件增加。

12.根据权利要求7所述的存储装置,其特征在于,所述第二确定单元,还用于获取所述将要合并的层中的前一层中文件字节最大的文件和预定文件,其中,所述预定文件为所述前一层中包含文件字节最大的文件的用户键的文件,所述用户键包含在与所述文件中的数据块对应的键中;基于所述文件字节最大的文件的用户键和所述预定文件的用户键,得到用户键范围;获取所述将要合并的层中的后一层中,用户键与所述用户键范围存在重叠的数据块,以及所述将要合并的层中的前一层中,用户键与所述用户键范围存在重叠的数据块;将重叠的数据块确定为将要修改的数据块。

13.一种存储系统,其特征在于,包括存储器、数据库和如权利要求7所述的存储装置,其中,所述存储器用于存储数据块和重组后的数据块,所述数据库用于存储所述数据块的键和统计信息、所述重组后的数据块的键和统计信息。

14.一种存储有计算机程序的计算机可读存储介质,其特征在于,当所述计算机程序被处理器执行时,实现如权利要求1至6中任一项所述的存储方法。

15.一种电子设备,其特征在于,包括:

至少一个处理器;

至少一个存储计算机可执行指令的存储器,

其中,所述计算机可执行指令在被所述至少一个处理器运行时,促使所述至少一个处理器执行如权利要求1到6中的任一权利要求所述的存储方法。

存储方法、存储装置和存储系统

技术领域

[0001] 本公开涉及数据存储技术,更具体地讲,涉及存储方法、存储装置和存储系统。

背景技术

[0002] 目前,存储和访问数百拍字节(PB)的数据是一个非常大的挑战,开源的RocksDB是用来存储和访问数百拍字节(PB)的数据的一种数据库。RocksDB是Facebook的开放的存储数据库,其是用C++编写的嵌入式的键值(Key-Value,简称为KV)型数据库,且RocksDB使用日志结构合并树(Log Structured Merge,简称为LSM)存储引擎存储数据,数据在存储设备上以排序序列表(Sorted Sequence Table,简称为SST)文件的形式存在。

[0003] 但是,相关技术中的RocksDB,在压缩(Compaction)过程中一个数据块发生了改变,会影响该SST文件中后面数据块的数据构成,甚至会影响下一个文件的数据块的数据构成,导致对大量已存在且未发生改变的数据的复写,极大的增加了写放大。

发明内容

[0004] 本公开提供了一种存储方法、存储装置和存储系统,以至少解决上述相关技术中的问题。

[0005] 根据本公开的一个方面,提供一种存储方法,包括:确定数据库中将要合并的层;基于数据库中将要合并的层的文件确定将要修改的数据块;根据将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组;在重组后的数据块的字节大小未超过第一预设值时,将存储器中将要修改的数据块替换为重组后的数据块;将重组后的数据块的键和统计信息存入数据库对应的文件。

[0006] 可选地,上述方法还包括:在重组后的数据块的字节大小超过第一预设值时,按第一预设值将重组后的数据块进行拆分;将存储器中的将要修改的数据块替换为拆分后的数据块;将重组后的数据块的键和统计信息存入数据库对应的文件包括:将拆分后的数据块的键和统计信息存入数据库对应的文件。通过本实施例,保证了一个数据块的字节大小不会过大。

[0007] 可选地,在将重组后的数据块的键和统计信息存入数据库对应的文件之后,还包括:判断对应的文件的所有数据块的字节大小是否超过第二预设值;在判断结果为是时,按所述第二预设值将对应的文件进行拆分,将重组后的数据块的键和统计信息以及对应的文件的未修改的数据块的键和统计信息,按键的顺序分别存入拆分后的文件;在判断结果为否时,将重组后的数据块的键和统计信息存入对应的文件。通过本实施例,保证了一个文件对应的总数据块的字节大小不会过大。

[0008] 可选地,确定数据库中将要合并的层包括:检测到数据库中的文件增加;获取数据库中每个层的字节大小;将每个层中字节大小超过对应层的字节阈值的层作为前一层;将前一层和后一层确定为将要合并的层,其中,后一层为前一层相邻的下一层。通过本实施例,可以方便快捷的触发压缩操作Compaction。

[0009] 可选地,检测到数据库中的文件增加包括:在写入的数据的字节大小超过第一预定阈值时,确定数据库中的文件增加。

[0010] 可选地,基于所述将要合并的层的文件确定将要修改的数据块包括:获取将要合并的层中的前一层中文件字节最大的文件和预定文件,其中,预定文件为所述前一层中包含文件字节最大的文件的用户键的文件,用户键包含在所述文件的数据块的键中;基于文件字节最大的文件的用户键和预定文件的用户键,得到用户键范围;获取将要合并的层中的后一层中,用户键与用户键范围存在重叠的数据块,以及将要合并的层中的前一层中,用户键与用户键范围存在重叠的数据块;将重叠的数据块确定为将要修改的数据块。通过本实施例,根据用户键重叠确定将要修改数据块,减少了重组过程对无关数据块进行重组。

[0011] 可选地,根据将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组包括:根据将要修改的数据块对应的用户键在存储器中读取对应的数据块,其中,用户键包含在数据块的键中;从所述对应的数据块中筛选出每一个用户键对应的最大序列号对应的数据,其中,所述序列号包含在所述数据块的键中,用于指示同一个用户键出现的顺序;将筛选出的数据按用户键进行排序后作为重组后的数据块。

[0012] 根据本公开示例性实施方式的另一个方面,提供一种存储装置,包括:第一确定单元,用于确定数据库中将要合并的层;第二确定单元,用于基于将要合并的层的文件确定将要修改的数据块确定;重组单元,用于根据将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组;替换单元,用于在重组后的数据块的字节大小未超过第一预设值时,将存储器中将要修改的数据块替换为重组后的数据块;写入单元,用于将重组后的数据块的键和统计信息存入数据库对应的文件。

[0013] 可选地,替换单元,还用于在重组后的数据块的字节大小超过第一预设值时,按第一预设值将重组后的数据块进行拆分;将存储器中的将要修改的数据块替换为拆分后的数据块;写入单元,还用于将拆分后的数据块的键和统计信息存入数据库对应的文件。

[0014] 可选地,写入单元,还用于在将重组后的数据块的键和统计信息存入数据库对应的文件之后,判断对应的文件的所有数据块的字节大小是否超过第二预设值;在判断结果为是时,按第二预设值将对应的文件进行拆分,将重组后的数据块的键和统计信息以及对应的文件的未修改的数据块的键和统计信息,按键的顺序分别存入拆分后的文件;在判断结果为否时,将重组后的数据块的键和统计信息存入对应的文件。

[0015] 可选地,第一确定单元,还用于检测到数据库中的文件增加;获取数据库中每个层的字节大小;将每个层中字节大小超过对应层的字节阈值的层作为前一层;将前一层和后一层确定为将要合并的层,其中,后一层为前一层相邻的下一层。

[0016] 可选地,第一确定单元,还用于在写入的数据的字节大小超过第一预定阈值时,确定数据库中的文件增加。

[0017] 可选地,第二确定单元,还用于获取将要合并的层中的前一层中文件字节最大的文件和预定文件,其中,预定文件为前一层中包含文件字节最大的文件的用户键的文件,用户键包含在所述文件中数据块的键中;基于文件字节最大的文件的用户键和预定文件的用户键,得到用户键范围;获取将要合并的层中的后一层中,用户键与用户键范围存在重叠的数据块,以及将要合并的层中的前一层中,用户键与用户键范围存在重叠的数据块;将重叠的数据块确定为将要修改的数据块。

[0018] 可选地,重组单元,还用于根据将要修改的数据块对应的用户键在存储器中读取对应的数据块,其中,用户键包含在数据块的键中;从所述对应的数据块中筛选出每一个用户键对应最大序列号对应的数据,其中,所述序列号包含在所述数据块的键中,用于指示同一个用户键出现的顺序;将筛选出的数据按用户键进行排序后作为重组后的数据块。

[0019] 根据本公开示例性实施方式的再一个方面,提供了一种存储系统,包括:括存储器、数据库和如上述实施例所述的存储装置,其中,存储器用于存储数据块和重组后的数据块,数据库用于存储数据块的键和统计信息、重组后的数据块的键和统计信息。

[0020] 根据本公开的另一个方面,提供一种存储有计算机程序的计算机可读存储介质,其中,当所述计算机程序被处理器执行时,实现本公开所述的部分或者全部存储方法。

[0021] 根据本公开的另一个方面,提供一种电子设备,包括:至少一个处理器;至少一个存储计算机可执行指令的存储器,其中,所述计算机可执行指令在被所述至少一个处理器运行时,促使所述至少一个处理器执行本公开所述的部分或者全部存储方法。

[0022] 本公开的技术方案提供了一种存储方法和存储装置,通过将原本存储在数据库的文件中的数据块存储在存储器中,而在数据库的文件中仅存储与数据块相关的信息用于后续调用和重组,由于存储器中存储的数据块大小是可变的,不再像存储在数据库中的数据块的大小是固定的,使得数据块的字节大小可以在一定范围内变化,从而数据库中文件对应的数据块实现可变长。

[0023] 将在接下来的描述中部分阐述本公开总体构思另外的方面和/或优点,还有一部分通过描述将是清楚的,或者可以经过本公开总体构思的实施而得知。

附图说明

[0024] 通过下面结合示例性地示出实施例的附图进行的描述,本公开的示例性实施例的上述和其他目的和特点将会变得更加清楚,其中:

[0025] 图1是示出相关技术中RocksDB的Compaction过程插入数据的示意图;

[0026] 图2是示出基于LSM的数据库系统各层写操作类型分布对比的示意图;

[0027] 图3是示出根据本公开的示例性实施方式的存储方法的流程图;

[0028] 图4是示出Block SSD和KV SSD架构对比的示意图;

[0029] 图5是示出根据本公开的示例性的实施例的可变长SST文件的格式示意图一;

[0030] 图6是示出根据本公开的示例性的实施例的可变长SST文件的格式示意图二;

[0031] 图7是示出根据本公开的示例性实施方式的可变长SST文件的Compaction过程插入数据的示意图;

[0032] 图8是示出根据本公开的示例性实施方式的基于KV SSD的可变长SST文件的Compaction的流程图;

[0033] 图9是示出根据本公开的示例性实施方式的基于LSM的数据库系统的最坏情况Compaction过程的示意图;

[0034] 图10是示出根据本公开的示例性实施方式的存储装置的框图。

具体实施方式

[0035] 提供下面的具体实施方式以帮助读者获得对在此描述的方法、设备和/或系统的

全面理解。然而,在理解本申请的公开之后,在此描述的方法、设备和/或系统的各种改变、修改和等同物将是清楚的。例如,在此描述的操作的顺序仅是示例,并且不限于在此阐述的那些顺序,而是除了必须以特定的顺序发生的操作之外,可如在理解本申请的公开之后将是清楚的那样被改变。此外,为了更加清楚和简明,本领域已知的特征的描述可被省略。

[0036] 在此描述的特征可以以不同的形式来实现,而不应被解释为限于在此描述的示例。相反,已提供在此描述的示例,以仅示出实现在此描述的方法、设备和/或系统的许多可行方式中的一些可行方式,所述许多可行方式在理解本申请的公开之后将是清楚的。

[0037] 在此使用的术语仅用于描述各种示例,并不将用于限制公开。除非上下文另外清楚地指示,否则单数形式也意在包括复数形式。术语“包含”、“包括”和“具有”说明存在叙述的特征、数量、操作、构件、元件和/或它们的组合,但不排除存在或添加一个或多个其他特征、数量、操作、构件、元件和/或它们的组合。

[0038] 除非另有定义,否则在此使用的所有术语(包括技术术语和科学术语)具有与由本公开所属领域的普通技术人员在理解本公开之后通常理解的含义相同的含义。除非在此明确地如此定义,否则术语(诸如,在通用词典中定义的术语)应被解释为具有与它们在相关领域的上下文和本公开中的含义一致的含义,并且不应被过于形式化地解释。

[0039] 此外,在示例的描述中,当认为公知的相关结构或功能的详细描述可能引起对本公开的模糊解释时,可省略这样的详细描述。

[0040] 相关技术中RocksDB的SST文件的数据块是固定大小的,一般为4K,Compaction过程中SST文件中一个数据块发生了改变,会影响该SST文件中后面数据块的数据构成,甚至会影响下一个文件的数据块的数据构成。如图1所示,Li+1层的第i个和第i+1个文件被选中参与Compaction,第i个文件中的数据块为[1 2 4 5],[6 7 8 9],[10 11 12 13]……[50 51 52 53],第i+1个文件中的数据块为[54 55 56 57],[58 59 60 61],[62 63 64 65]……[94 95 96 97]。SST文件是有序的,数据3应该被插入到第i个文件的第1个数据块,第i个文件的第一个数据块就变成了[1 2 3 4],由于SST文件的块的是固定大小的,重组后数据5就出现在第i个文件的第二个数据块,依次类推,第i个文件的最后一个数据可能就出现在第i+1个文件的第一个数据块,导致第i+1个文件的数据发生了变化。而且,Compaction过程是以SST文件为单位的,即所有参与Compaction的Li层和Li+1层的文件都会进行读取、解析、排序、重组和复写。

[0041] 图2是示出基于LSM的数据库系统各层写操作类型分布对比的示意图,图2中的基于LSM的数据库系统包括标记为S1、S2和S3的三个数据库系统。

[0042] 如图2所示,通过对基于日志结构合并树(LSM)的三种数据库系统各层磁盘的写操作进行了分析,L0层的文件中的数据来自主存,只有新数据的写入;L1～L4层的文件是由Compaction操作生成的,可以看出,基于LSM的数据库系统的L1～L4层的Compaction中的写操作的消耗主要来源于对大量已存在且未发生改变的数据的拷贝,极大的增加了写放大。

[0043] 针对上述问题,根据本公开示例性实施方式的一个方面,提供一种存储方法。图3是示出根据本公开的示例性实施方式的存储方法的流程图。该存储方法应用于上述实施例中的存储设备上,任何存储方法都可以作为由电子设备部分或完全实现的存储过程来执行:

[0044] 在步骤S301中,确定数据库中将要合并的层。上述数据库可以为键值KV型数据库,

例如,可以为RocksDB。需要说明的是,图3的存储方法中使用的数据库并不限于上述数据库。

[0045] 在本公开的实施例中,确定数据库中将要合并的层可以包括:检测到数据库中的文件的字节大小增加;获取数据库中每个层的字节大小;将每个层中字节大小超过对应层的字节阈值的层作为前一层;将前一层和后一层确定为将要合并的层,其中,后一层为前一层相邻的下一层。通过寻址和/或通过创建后一层和前一层的时序,后一层可以逻辑上与前一层相邻,和/或后一层可以在数据库的存储位置中物理上相邻。通过本实施例,可以方便快捷的触发压缩操作Compaction。需要说明的是,上述字节阈值可以存储在每层的数据块的统计信息中。

[0046] 例如,检测到写入的数据达到一定值(如,第一预定阈值),则需要在数据库中增加SST文件以便存储写入的数据,此时,可以从统计信息中获取数据库中每个层的字节大小。每个层可以具有特定于该层的字节阈值。当任一层的字节大小超过对应层的字节阈值的情况下,则将字节大小超过对应层的字节阈值的层即Li层,该层相邻的下一层即Li+1层。可选地,可以先判断写入数据的文件对应的层,即逻辑树的第0层,先比较第0层的当前字节总数与该层的字节阈值,在当前字节总数大于该层的字节阈值,则第0层即Li层,该层的相邻下一层即为,也即第1层为Li+1层,完成一次compaction后,可以继续判断第1层的当前字节总数与该层的字节阈值,如果当前字节总数大于该层的字节阈值,则第1层即Li层,该层的相邻下一层即为,也即第2层为Li+1层,如果当前字节总数不大于该层的字节阈值,可以停止compaction。

[0047] 在该实施例中,参与合并的所有Li层和Li+1层的文件都会在合适的时机被后台线程删除,删除的先后顺序可以是compaction的顺序,也可以是根据需要设定的顺序。

[0048] 另外,在合并完第一次后,确定合并的层还可以根据如下方式,删除部分SST文件时,也可以根据删除的文件对应的层的当前字节总数与所述对应的层的字节阈值确定将要合并的Li层和Li+1层。需要说明的,数据库中每一层对应有自己的字节阈值,字节阈值可以根据实际需要动态的设定。

[0049] 在本公开的实施例中,当任何层的写入的数据的字节大小超过该层的第一预定阈值时,可以检测到数据库中的文件的数量增加。也就是说,基于层的写入数据的字节大小超过该层的第一预定阈值,可以检测到数据库中的文件的数据增加。在步骤S302中,基于数据库中将要合并的层的文件确定将要修改的数据块。

[0050] 在本公开的实施例中,基于所述将要合并的层的文件确定将要修改的数据块可以包括:获取将要合并的层中的前一层中文件字节最大的文件和预定文件,其中,预定文件为前一层中包含文件字节最大的文件的用户键的文件,用户键包含在与文件中的数据块对应的键中。可基于文件字节最大的文件的用户键和预定文件的用户键,得到用户键范围。可获取将要合并的层中的后一层中,用户键与用户键范围存在重叠的数据块,以及将要合并的层中的前一层中,用户键与用户键范围存在重叠的数据块。可将重叠的数据块确定为将要修改的数据块。通过本实施例,根据用户键重叠确定将要修改数据块,减少了重组过程对无关数据块进行重组。

[0051] 例如,在确定将要合并的Li层和Li+1层后,可以先对Li层的文件按字节大小进行排序,得到排序第一的文件,将排序第一的文件与Li层的其他文件进行比对,得到包含排序

第一的文件的用户键的文件,即得到上面的预定文件。再基于排序第一的文件的用户键和预定文件的用户键得到一个用户键范围,在Li+1层的文件中查询用户键与该用户键范围有重叠的数据块,以及在Li层的文件中查询用户键与该用户键范围有重叠的数据块,该有重叠的数据块即为将要修改的数据块。

[0052] 在步骤S303中,图3所示的方法包括根据将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组。上述存储器可以为直接支持KV存储的固态硬盘,例如,固态硬盘可以是键值固态硬盘KV SSD。该存储器并不限于上述提到的固态硬盘,只要能存储数据块且存储的数据块的字节大小是可变的均可以应用在本公开中。上述键可以包括但不限于:数据块的用户键和序列号,序列号用于指示同一个用户键出现的顺序,用户键是面向用户的,用户在调用数据块时需要使用的键。

[0053] 在本公开的实施例中,可如下根据将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组可根据将要修改的数据块对应的用户键在存储器中读取对应的数据块。用户键包含在与数据块对应的键中。可从对应的数据块中筛选出每一个用户键对应的最大序列号对应的数据。序列号包含在所述数据块的键中,用于指示同一个用户键出现的顺序。将筛选出的数据按用户键进行排序后作为重组后的数据块。

[0054] 例如,对于将要修改的数据块,可以根据数据块的用户键从KV SSD读取出来,从读取的数据块中查询同一个用户键对应的数据。仅保留该同一个用户键对应的数据中最大序列号对应的数据。删除其他序列号对应的数据,然后,将保留的数据按用户键进行排序,组成重组后的数据块。

[0055] 在步骤S304中,在重组后的数据块的字节大小未超过第一预设值时,将存储器中将要修改的数据块替换为重组后的数据块。由于数据块的字节大小在一定范围内可变,因此,如果重组后的数据块没有超出数据块长度阈值(如下面的第一预设值),则对将要修改的数据块进行重组后,可以直接存储到存储器的对应的位置。不需要改变的数据块可以原样保留在KV SSD中,只将该数据块的元信息全部存储在可变长的SST文件中,上述第一预设值可以根据实际需要动态设定。

[0056] 在步骤S305中,将重组后的数据块的键和统计信息存入数据库对应的文件。

[0057] 在本公开的这些实施例中,在重组后的数据块的字节大小超过第一预设值时,按第一预设值将重组后的数据块进行拆分。将存储器中的将要修改的数据块替换为拆分后的数据块。也就是说,重组后的数据块根据第一预设值被拆分,是基于(例如上述条件)重组后的数据块的字节大小超过第一预设值。将重组后的数据块的键和统计信息存入数据库对应的文件包括:将拆分后的数据块的键和统计信息存入数据库对应的文件。通过本实施例,保证了一个数据块的字节大小不会过大。

[0058] 例如,如果重组后的数据块超出数据块长度阈值(即第一预设值),此时会把重组的数据块按长度阈值分成多个数据块,并将该多个数据块分别写入KV SSD,并将其对应的元信息存储在可变长的SST文件中,对于不需要改变的数据块,可以将其原样保留在KV SSD中,只将该数据块的元信息全部存储在可变长的SST文件中。上述拆分过程基于第一预设值和重组数据块的字节大小,可以如下:假设第一预设值为16,重组后的数据块的字节大小为33,此时重组后的数据块可以拆分为16、16、1这样三个大小的数据块。

[0059] 在本公开的实施例中,在将重组后的数据块的键和统计信息存入数据库对应的文

件之后,还可以判断对应的文件的所有数据块的字节大小是否超过第二预设值;在判断结果为是时,按第二预设值将对应的文件进行拆分,将重组后的数据块的键和统计信息以及对应的文件的未修改的数据块的键和统计信息,按键的顺序分别存入拆分后的文件;在判断结果为否时,将重组后的数据块的键和统计信息存入对应的文件。上述第二预设值可以根据实际将要动态设定。因此,处理方式根据判断结果而变化。通过本实施例,保证了一个文件对应的总数据块的字节大小不会过大。

[0060] 例如,如果将重组的数据块的相关信息存储在SST文件后,可变长的SST文件对应的数据块的总字节大小超过一定范围,则可以将可变长的SST文件分成多个文件,如假设第二预设值为100,对应的文件的字节大小为103,此时对应的文件可以拆分为100、3这样两个大小的文件。通过这种方式可以保证每个文件对应的总数据块的字节大小不会过大。

[0061] 通过本公开的上述实施例,Compaction过程中细化粒度,把以SST文件为单位变为以数据块为单位进行Compaction。对于Compaction过程中发生改变的数据块,对该数据块进行读取、解析、排序、重组和复写,使该数据块在一定范围内大小可变。对于Compaction过程中没有发生改变的数据块,保持原样存储在KV SSD中,不受变化了的数据块的影响。这样能减少不必要的操作和IO消耗,从而降低写放大,提高写性能。

[0062] 下面以数据库为RocksDB、存储器为KV SSD为例,结合图4、图5、图6、图7和图8详细说明Compaction过程。

[0063] 图4是示出Block SSD和KV SSD架构对比的示意图,Block SSD架构如图4的左侧所示,KV SSD架构如图4的右侧所示,上述提到的KV SSD是新型的SSD,可以直接支持KV存储的设备。相对于相关技术中Block型SSD,其KV数据库的写入和读取,需要完成从KV到文件,文件到逻辑区块地址(Logical Block Address,简称为LBA),再从LBA到物理区块地址(Physics Block Address,简称为PBA)的数据转换。而KV SSD采用了一种增强的闪存转换层(Flash Translation Layer,简称为FTL),实现了KV存储的部分核心功能,其向外提供KV接口,能够直接响应主机端应用程序的KV请求,简化了数据读写的流程,不但提高了数据读写的效率,还大大减少了主机端CPU和内存的消耗。

[0064] 而将KV SSD与KV存储引擎或KV数据库(如RocksDB)配合使用,在诸多方面都会带来较大的提升。因此,可以将RocksDB的SST文件的数据块存储在KV SSD中,一是利用KV SSD的优势,即,当数据块大小在一定范围内变化时,KV SSD的随机读写数据的性能稳定,能保证RocksDB的读写性能,如图5所示,通过KV SSD实现存储字节长度可变的数据块,实现可变长的SST文件;二是通过KV SSD实现存储字节长度可变的数据块,让SST文件中各个数据块相互独立,读写互不影响,便于后续的Compaction过程中细化粒度。

[0065] 在本实施例中,上述数据块的键和数据块的统计信息可以存储在文件的元信息的索引块中。即可以对元信息中的索引块的数据结构进行扩充,增加了键和数据块的统计信息,在沿用原有结构的基础上,方便快速的存储键和数据块的统计信息。具体地,如图6所示,对SST文件中元信息中的数据索引块的数据结构进行扩充,在原来的基础上,增加了KV SSD中key的相关字段(SST文件的文件编号和数据块在SST文件中序列号)和数据块统计信息,且增加的KV SSD中key的相关字段和数据块统计信息分别在读写KV SSD和Compaction过程中使用。

[0066] 图7是示出根据本公开的示例性实施方式的可变长SST文件的Compaction过程插

入数据的示意图,如图7所示,在Compaction过程中,需插入数据3的时候,数据块[1 2 4 5]变成[1 2 3 4 5],其他数据块保持不变。

[0067] 图8是示出根据本公开的示例性实施方式的基于KV SSD的可变长SST文件的Compaction的流程图,如图8所示,Compaction过程如下:

[0068] 在步骤S810,选取将要合并的Li层和Li+1层的SST文件。在该过程中,可以先检测到写入的数据的字节大小,在字节大小达到一定值(上述第一预设阈值),则确定数据库中增加了SST文件,此时可以确定写入数据的层以及该层相邻的下一层作为将要合并的Li层和Li+1层;也可以监测增加的文件对应的层当前字节大小与该层的字节阈值关系,在增加的文件对应的层当前字节大小大于该层的字节阈值时,将增加的文件对应的层以及该层相邻的下一层作为将要合并的层Li和Li+1层;

[0069] 在步骤S820,判断SST文件中的数据块是否需要修改。在该过程中,对Li层的文件按字节大小进行排序,得到排序第一的文件,以及Li层中用户键与排序第一的文件的用户键重叠的文件(即上述预定文件),基于排序第一的文件的用户键和预定文件的用户键,得到一个用户键范围,在Li+1层中查询与该用户键范围有重叠的数据块以及在Li层中查询与该用户键范围有重叠的数据块,则将有重叠的数据块确定为将要修改的数据块;

[0070] 再有,在步骤S830,从KV SSD中读取将要修改的数据块,并进行解析、排序、重组。

[0071] 然后,在步骤S840,判断重组后的数据块的字节大小是否超过数据块阈值(上述第一预设值),该数据块阈值可以存储在数据块的统计信息中;

[0072] 再然后,在步骤S850,在重组后的数据块的字节大小超过数据块阈值时(即在S840判断结果为是时),将重组的数据块按数据块阈值拆分为若干个数据块,然后,在步骤S860,将若干个数据块存入KV SSD并将若干个数据块的相关信息写入可变长的SST文件中;在重组后的数据块的字节大小没有超过阈值时(即在S840判断结果为否时),直接执行步骤S860,即直接将重组的数据块存入KV SSD并将重组的数据块的相关信息写入可变长的SST文件中;

[0073] 完成上述操作后,在步骤S870,再判断写入后的可变长SST文件对应的所有的数据块的总字节大小是否超过文件阈值(上述第二预设值),如果超过文件阈值(上述第二预设值)(即在S870判断结果为是时),如步骤S880,则将SST文件按文件阈值拆分为若干个文件,可选地,也可以关闭当前可变长SST文件,将重组的数据块的元信息写入下一个可变长的SST文件,如果没有超过文件阈值(即在S870判断结果为否时),如步骤S890,则将重组的数据块的元信息写入当前可变长的SST文件。

[0074] 上述参与compaction的所有文件都会被删除,重组的数据块都会存在未参与compaction的文件中,如上面提到的重组的数据块的元信息可写入下一个可变长的SST文件,也可以存储在新的SST文件中。删除文件后可重新计算对应层当前字节总数,并与该层的字节阈值比较,得到比较结果,方便下次compaction用。

[0075] 为了验证上述实施例的有效性,本公开通过计算LSM存储引擎在Compaction过程的写放大,以最坏情况的Compaction为例(在L0层的文件,通过Compaction一直落到最后一层)说明上述实施例降低Compaction写放大的情况。表1给出了需要用到的基本定义,图9给出了基于LSM的数据库系统的最坏情况Compaction过程。

[0076] 表1基本定义

[0077]

项目	定义	单位

[0078]

N	一个 SST 文件中的数据块的个数	个
T	每层之间的总数据量的倍数	
L	LSM 树的层数	

[0079] L0层的文件达到Compaction的触发条件,跟L1层的文件进行Compaction,L0层的数据落到L1层,根据LSM树的特点,下一层的总容量是上一层总容量的T倍,最坏的情况L0层跟L1层将要进行T次Compaction,L1层的数据才会达到与L2层Compaction的条件,落到L2层;同样的最坏的情况下,L1层将要与L2层的数据进行T次Compaction,才会触发L2层与L3层Compaction的条件,使数据落到L3层,以此类推,L0层的数据落到L层的时候,已经进行了T×L次Compaction,分摊到每个文件的每个数据块,RocksDB的Compaction过程中的写放大为:

[0080] $$O\left(\frac{T}{N}\right)+O\left(\frac{T}{N}\right)+\cdots+O\left(\frac{T}{N}\right)=O\left(\frac{T*L}{N}\right)$$

[0081] 对于可变长的SST文件,每次Compaction,平均只有1/T的数据块会发生改变,本公开的写放大为:

[0082] $$O\left(\frac{T}{N}*\frac{1}{T}\right)+O\left(\frac{T}{N}*\frac{1}{T}\right)+\cdots+O\left(\frac{T}{N}*\frac{1}{T}\right)=O\left(\frac{L}{N}\right)$$

[0083] 通过理论分析,本公开的写放大会比原来基于LSM的数据库系统的写放大减小T倍。

[0084] 根据本公开示例性实施方式的另一个方面,提供一种存储装置,应用于上述实施例的存储设备上,参照图10,该存储装置包括:第一确定单元102、第二确定单元104、重组单元106、替换单元108和写入单元1010。

[0085] 在继续进行之前,应当清楚的是,本文中的附图(包括图10)示出并引用具有诸如“单元”、“电路”或“块”之类的标签的电路。一般地,如在本文中所描述的(一个或多个)发明构思的领域中,可以根据执行所描述的一个或多个功能的单元、电路和块来描述和示出示例。这些单元、电路和块(其在本文中可以被称为第一确定单元、第二确定单元、重组单元、替换单元、写入单元等)由模拟和/或数字电路(诸如逻辑门、集成电路、微处理器、微控制器、存储器电路、无源电子组件、有源电子组件、光学组件、硬连线电路等)物理的实现,并且可以可选地由硬件和/或软件驱动。例如,电路可以体现在一个或多个半导体芯片中,或者体现在诸如印刷电路板等的基板支撑件上。构成单元、电路或块的电路可以由专用硬件实现,或者由处理器(例如,一个或多个编程的微处理器和相关联的电路)实现,或者由执行单元、电路或块的一些功能的专用硬件和执行单元、电路或块的其他功能的处理器的组合实现。在不脱离本公开的范围的情况下,示例的每个单元、电路或块可以物理地分成两个或更多个可交互和分立的电路或块。同样地,在不脱离本公开的范围的情况下,示例的单元、电路和块可以物理地组合成更复杂的单元、电路和块。

[0086] 第一确定单元102,用于确定数据库中将要合并的层;第二确定单元104,用于基于

将要合并的层的文件确定将要修改的数据块;重组单元106,用于根据将要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组;替换单元108,用于在重组后的数据块的字节大小未超过第一预设值时,将存储器中将要修改的数据块替换为重组后的数据块;写入单元1010,用于将重组后的数据块的键和统计信息存入数据库对应的文件。

[0087] 在本公开的实施例中,替换单元108,还用于在重组后的数据块的字节大小超过第一预设值时,按第一预设值将重组后的数据块进行拆分;将存储器中的将要修改的数据块替换为拆分后的数据块;写入单元1010,还用于将拆分后的数据块的键和统计信息存入数据库对应的文件。

[0088] 在本公开的实施例中,写入单元1010,还用于在将重组后的数据块的键和统计信息存入数据库对应的文件之后,判断对应的文件的所有数据块的字节大小是否超过第二预设值;在判断结果为是时,按第二预设值将对应的文件进行拆分,将重组后的数据块的键和统计信息以及对应的文件的未修改的数据块的键和统计信息,按键的顺序分别存入拆分后的文件;在判断结果为否时,将重组后的数据块的键和统计信息存入对应的文件。

[0089] 在本公开的实施例中,第一确定单元102,还用于检测到数据库中的文件增加;获取数据库中每个层的字节大小;将每个层中字节大小超过对应层的字节阈值的层作为前一层;将前一层和后一层确定为将要合并的层,其中,后一层为前一层相邻的下一层。

[0090] 在本公开的实施例中,第一确定单元102,还用于在写入的数据的字节大小超过第一预定阈值时,确定数据库中的文件增加。

[0091] 在本公开的实施例中,第二确定单元104,还用于获取将要合并的层中的前一层中文件字节最大的文件和预定文件,其中,预定文件为前一层中包含文件字节最大的文件的用户键的文件,用户键包含在所述文件中数据块的键中;基于文件字节最大的文件的用户键和预定文件的用户键,得到用户键范围;获取将要合并的层中的后一层中,用户键与用户键范围存在重叠的数据块,以及将要合并的层中的前一层中,用户键与用户键范围存在重叠的数据块;将重叠的数据块确定为将要修改的数据块。

[0092] 在本公开的实施例中,重组单元106,还用于根据将要修改的数据块对应的用户键在存储器中读取对应的数据块,其中,用户键包含在数据块的键中;从所述对应的数据块中筛选出每一个用户键对应的最大序列号对应的数据,其中,所述序列号包含在数据块的键中,用于指示同一个用户键出现的顺序;将筛选出的数据按用户键进行排序后作为重组后的数据块。

[0093] 根据本公开示例性实施方式的再一个方面,提供了一种存储系统,包括:存储器、数据库和如上述实施例所述的存储装置,其中,存储器用于存储数据块和重组后的数据块,数据库用于存储数据块的键和统计信息、重组后的数据块的键和统计信息。统计信息包括对数据块进行重组的相关信息,如字节大小、字节阈值等等。

[0094] 本公开的上述实施例给出了一种优选存储系统和存储方法,如,上述实施例提供了基于新型存储硬件KV SSD对RocksDB进行优化的方案,该技术方案可以把RocksDB的SST文件中的数据块存储在KV SSD中,当数据大小在一定范围内变化时,KV SSD随机读写数据的性能稳定,能保证RocksDB的读写性能,实现可变长的SST文件;而且,细化RocksDB的Compaction过程的粒度,把以文件为单位细化为以数据块为单位,避免Compaction过程中

因为SST文件中部分数据块的改变,而对所有数据块进行读取,解析,重组和复写,减少不必要的操作和IO消耗,从而降低写放大,提高写性能。

[0095] 另外,本公开描述了优化,即能够把RocksDB的Compaction过程中的写放大从$O\left(\frac{T*L}{N}\right)$降到$O\left(\frac{L}{N}\right)$,降低T倍。

[0096] 应该理解,根据本公开的示例性实施方式的存储方法和存储装置中的各个单元/模块可被实现为硬件组件和/或软件组件。本领域技术人员根据限定的各个单元/模块所执行的处理,可以例如使用现场可编程门阵列(FPGA)或专用集成电路(ASIC)来实现各个单元/模块。

[0097] 根据本公开示例性实施例的再一个方面,提供一种存储有计算机程序的计算机可读存储介质,其中,当所述计算机程序被处理器执行时,实现本公开所述的存储方法。

[0098] 具体地,根据本公开的示例性实施例的存储方法可被编写为计算机程序、代码段、指令或它们的任何组合,并且任何计算机程序、代码段、指令或其组合可以被记录、存储或固定在一个或多个非暂时性计算机可读存储介质中或一个或多个非暂时性计算机可读存储介质上。所述计算机可读存储介质是可存储由计算机系统读出的数据的任意数据存储装置。计算机可读存储介质的示例包括:只读存储器、随机存取存储器、只读光盘、磁带、软盘、光数据存储装置和通过互联网经有线或无线的传输路径。

[0099] 根据本公开示例性实施例的又一个方面,提供一种电子设备,包括:至少一个处理器;至少一个存储计算机可执行指令的存储器,其中,所述计算机可执行指令在被所述至少一个处理器运行时,促使所述至少一个处理器执行本公开所述的存储方法。

[0100] 具体地,所述电子设备可以广义地为平板电脑、智能手机、智能手表,或任何其他具有必要的计算和/或处理能力的电子设备。在一个实施例中,该电子设备可包括通过系统总线连接的处理器、存储器、网络接口、通信接口等。该电子设备的处理器可用于提供必要的计算、处理和/或控制能力。该电子设备的存储器可包括非易失性存储介质和内存储器。该非易失性存储介质中或上可存储有操作系统、计算机程序等。该内存储器可为非易失性存储介质中的操作系统和计算机程序的运行提供环境。该电子设备的网络接口和通信接口可用于与外部的设备通过网络连接和通信。

[0101] 上面已经描述了本公开的示例性实施例,并且应当理解,前面的描述仅是示例性的而非穷举性的,并且本公开不限于所公开的示例性实施例。在不脱离本公开的范围和精神的情况下,许多修改和改变对于本领域普通技术人员而言是显而易见的。因此,本公开的保护范围应当服从权利要求的范围。

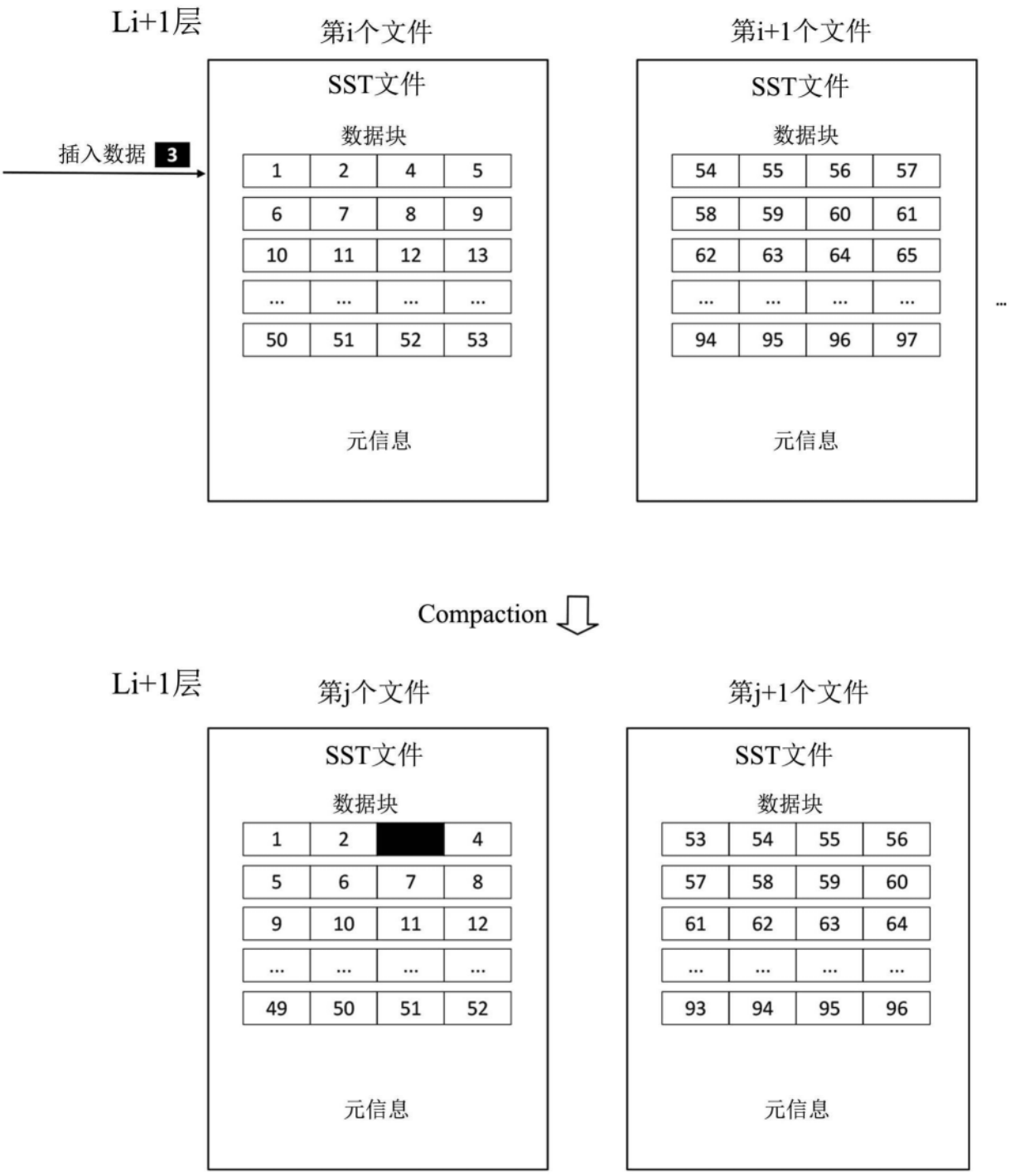

Li+1层
第i个文件
第i+1个文件
SST文件
数据块
插入数据 3
1 2 4 5
6 7 8 9
10 11 12 13
... ...
50 51 52 53
元信息
SST文件
数据块
54 55 56 57
58 59 60 61
62 63 64 65
... ...
94 95 96 97
元信息
...
Compaction
Li+1层
第j个文件
第j+1个文件
SST文件
数据块
1 2 4
5 6 7 8
9 10 11 12
... ...
49 50 51 52
元信息
SST文件
数据块
53 54 55 56
57 58 59 60
61 62 63 64
... ...
93 94 95 96
元信息

图1

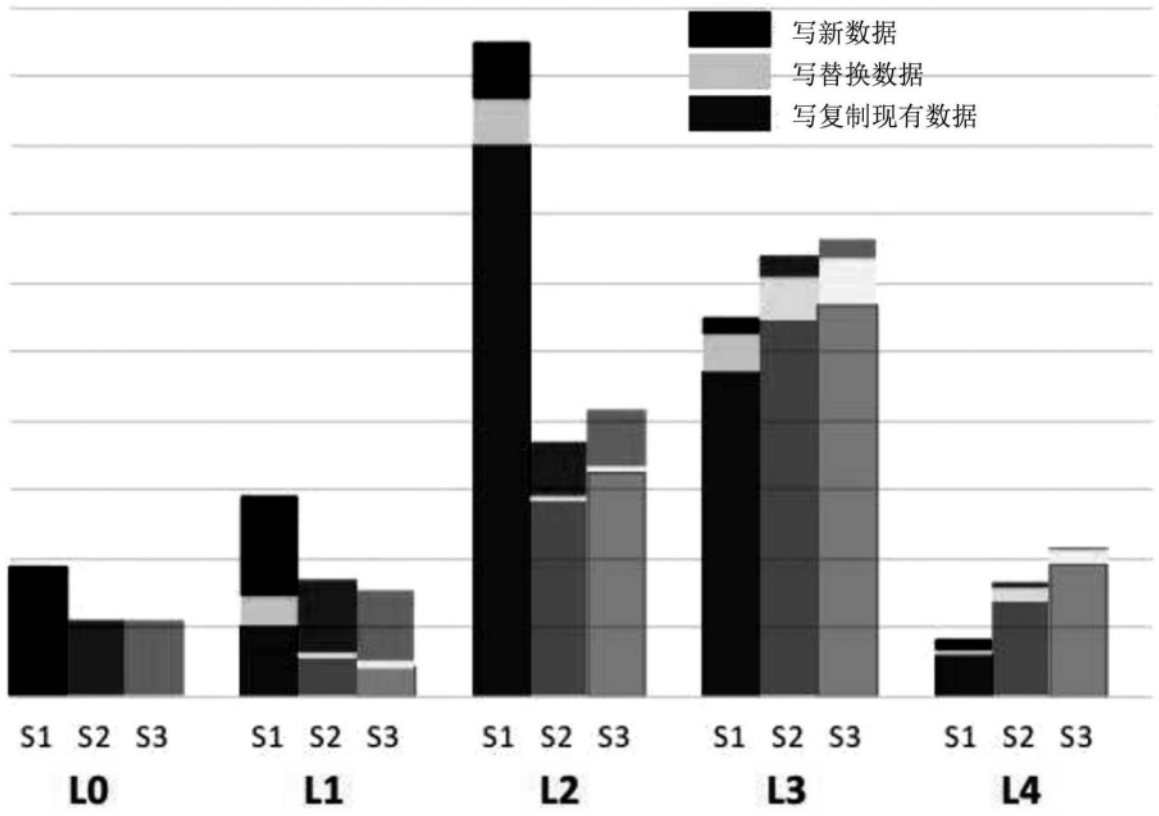

写新数据
写替换数据
写复制现有数据
S1 S2 S3
L0
S1 S2 S3
L1
S1 S2 S3
L2
S1 S2 S3
L3
S1 S2 S3
L4

图2

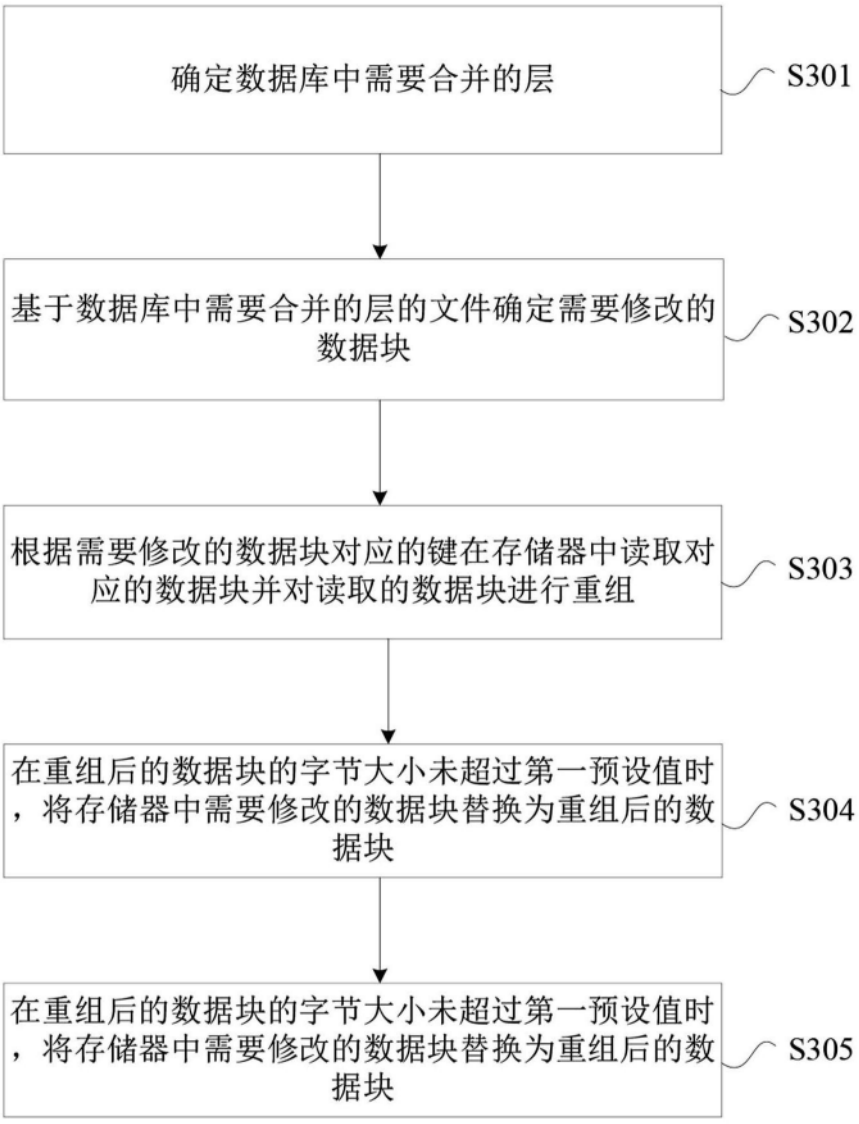

确定数据库中需要合并的层
S301
基于数据库中需要合并的层的文件确定需要修改的数据块
S302
根据需要修改的数据块对应的键在存储器中读取对应的数据块并对读取的数据块进行重组
S303
在重组后的数据块的字节大小未超过第一预设值时，将存储器中需要修改的数据块替换为重组后的数据块
S304
在重组后的数据块的字节大小未超过第一预设值时，将存储器中需要修改的数据块替换为重组后的数据块
S305

图3

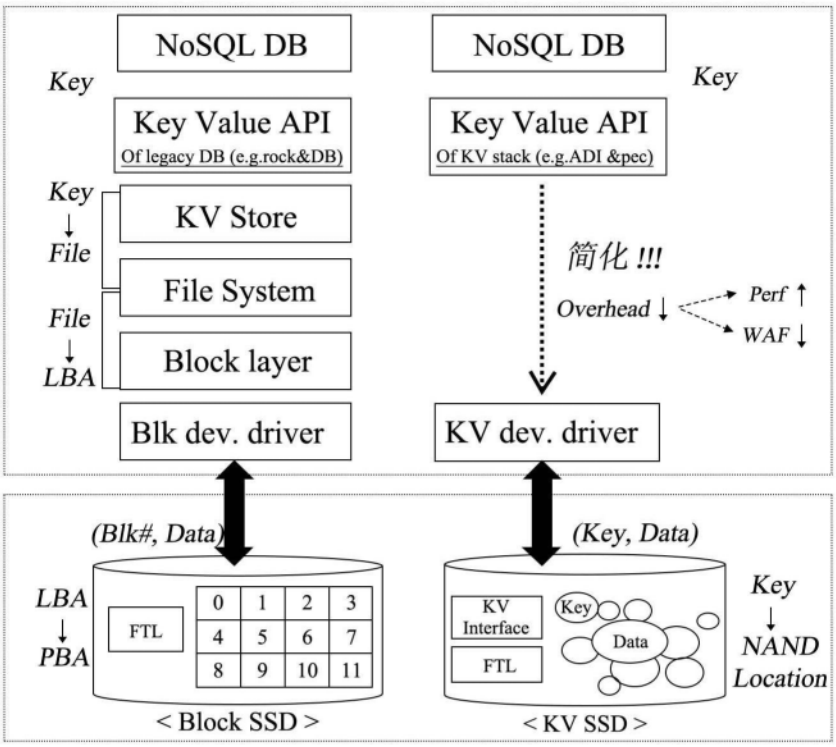

NoSQL DB
NoSQL DB
Key
Key
Key Value API
Of legacy DB (e.g.rock&DB)
Key Value API
Of KV stack (e.g.ADI &pec)
Key
File
KV Store
File System
File
LBA
Block layer
Blk dev. driver
KV dev. driver
简化 !!!
Overhead
Perf
WAF
(Blk#, Data)
(Key, Data)
LBA
PBA
FTL
0 1 2 3
4 5 6 7
8 9 10 11
KV Interface
FTL
Key
Data
Key
NAND Location
< Block SSD >
< KV SSD >

图4

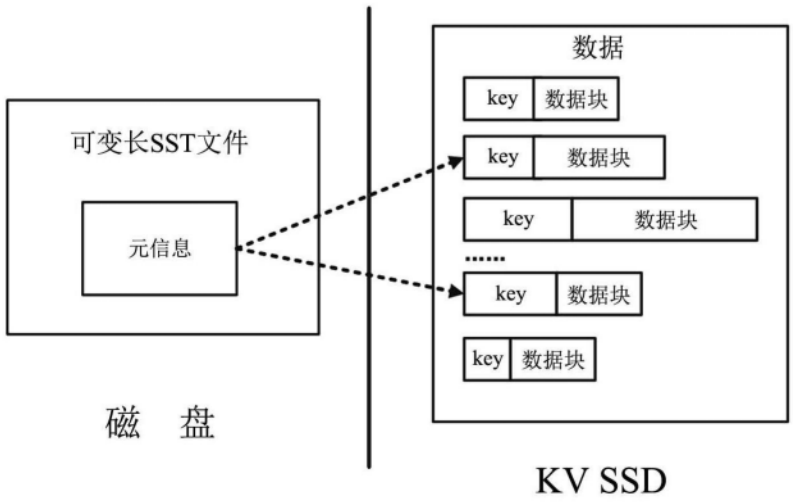

可变长SST文件
元信息
磁 盘
数据
key 数据块
key 数据块
key 数据块
key 数据块
key 数据块
KV SSD

图5

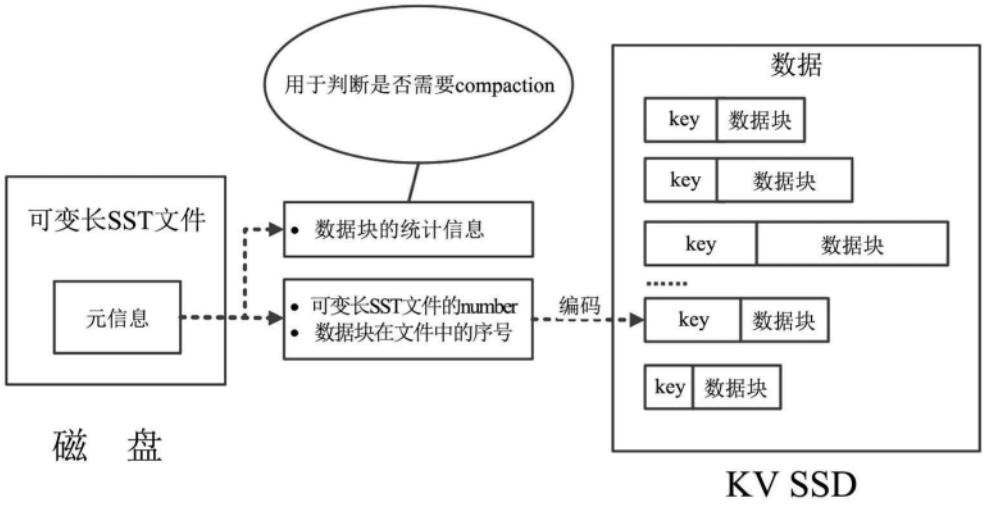

用于判断是否需要compaction
可变长SST文件
元信息
磁 盘
数据块的统计信息
可变长SST文件的number
数据块在文件中的序号
编码
数据
key
数据块
key
数据块
key
数据块
key
数据块
key
数据块
KV SSD

图6

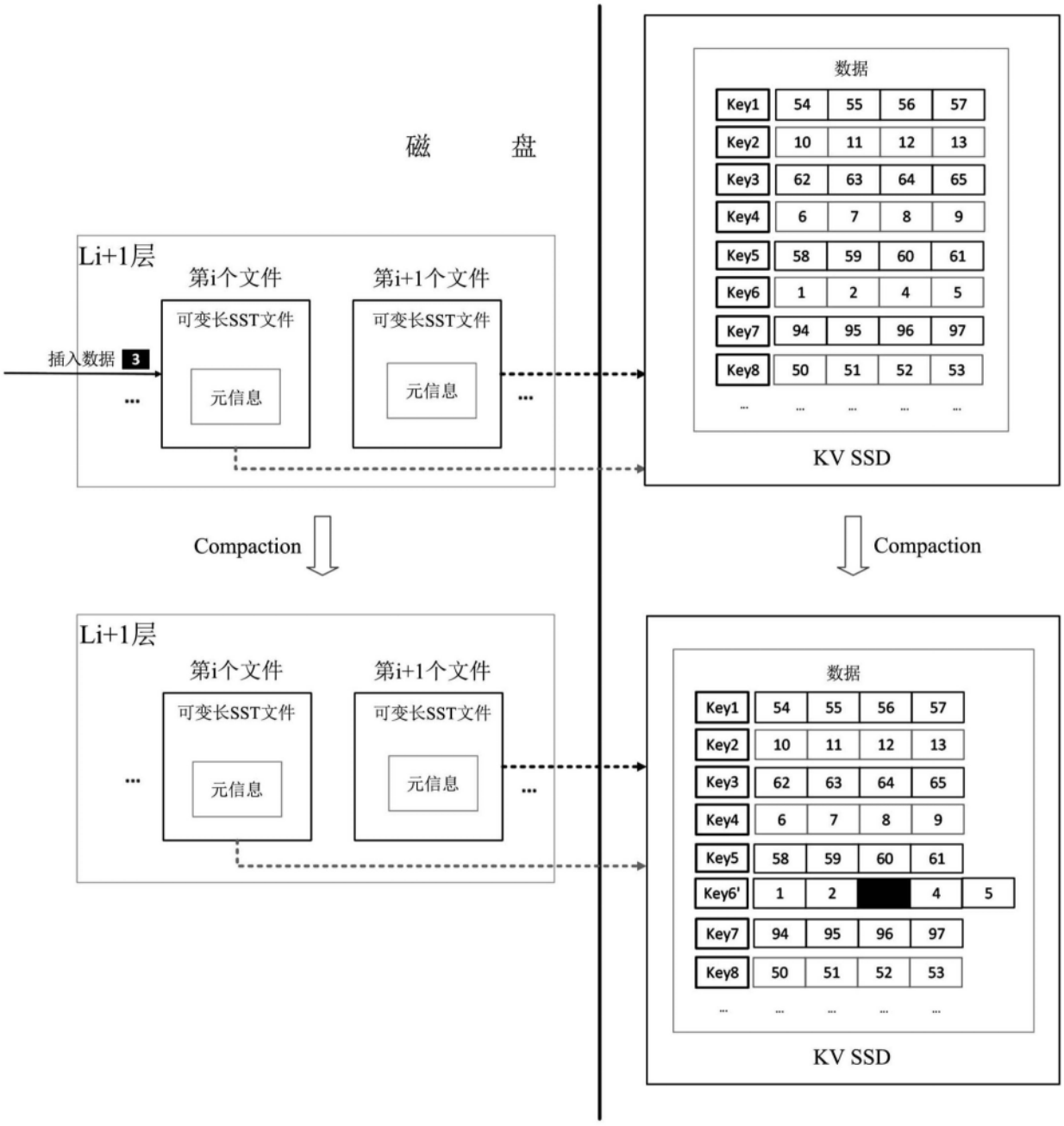

磁 盘
Li+1层
第i个文件
第i+1个文件
可变长SST文件
可变长SST文件
插入数据 3
元信息
元信息
数据
Key1 54 55 56 57
Key2 10 11 12 13
Key3 62 63 64 65
Key4 6 7 8 9
Key5 58 59 60 61
Key6 1 2 4 5
Key7 94 95 96 97
Key8 50 51 52 53
KV SSD
Compaction
Compaction
Li+1层
第i个文件
第i+1个文件
可变长SST文件
可变长SST文件
元信息
元信息
数据
Key1 54 55 56 57
Key2 10 11 12 13
Key3 62 63 64 65
Key4 6 7 8 9
Key5 58 59 60 61
Key6' 1 2 4 5
Key7 94 95 96 97
Key8 50 51 52 53
KV SSD

图7

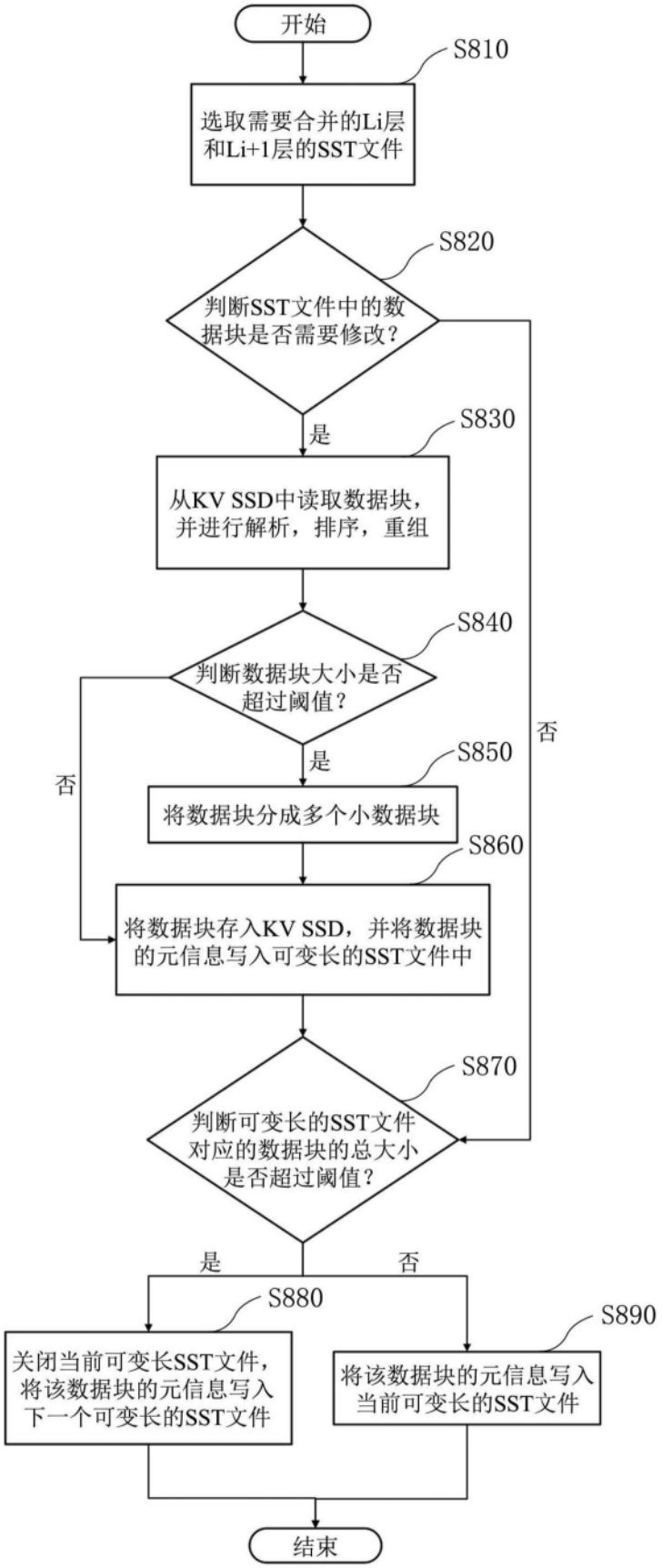

开始
S810
选取需要合并的Li层和Li+1层的SST文件
S820
判断SST文件中的数据块是否需要修改？
是
否
S830
从KV SSD中读取数据块，并进行解析，排序，重组
S840
判断数据块大小是否超过阈值？
是
否
S850
将数据块分成多个小数据块
S860
将数据块存入KV SSD，并将数据块的元信息写入可变长的SST文件中
S870
判断可变长的SST文件对应的数据块的总大小是否超过阈值？
是
否
S880
关闭当前可变长SST文件，将该数据块的元信息写入下一个可变长的SST文件
S890
将该数据块的元信息写入当前可变长的SST文件
结束

图8

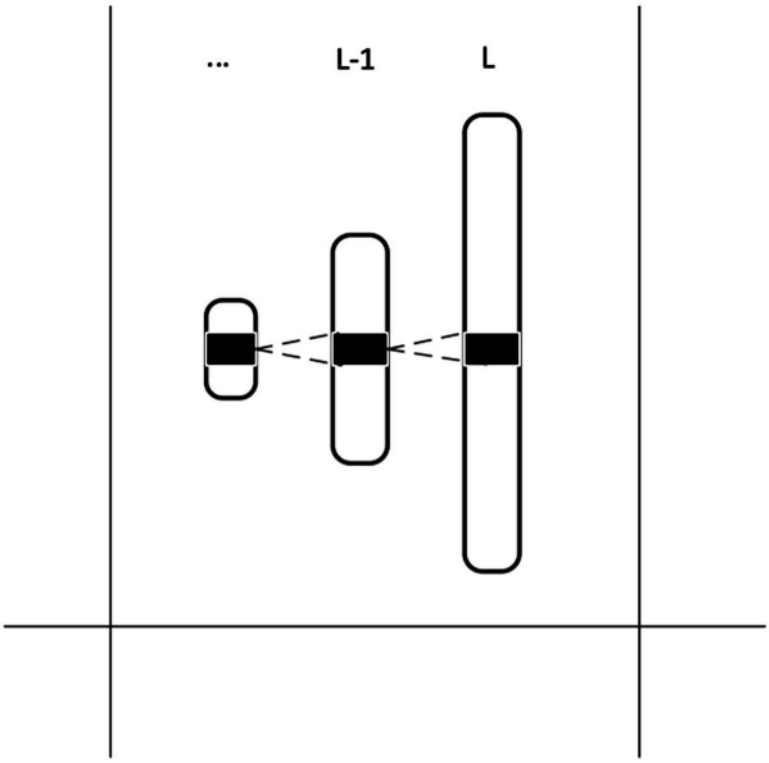


图9

图10