[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 3/06 (2006.01)
G06F 12/00 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200410058582.5

[43] 公开日 2006 年 2 月 22 日

[11] 公开号 CN 1737745A

[22] 申请日 2004.8.18
[21] 申请号 200410058582.5
[71] 申请人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼
[72] 发明人 张 巍 唐小松 黄玉环 张国彬 张 粤 任雷鸣 陈绍元

[74] 专利代理机构 北京德琦知识产权代理有限公司
代理人 张颖玲 王 琦

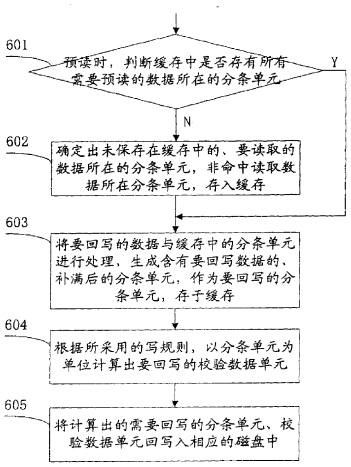
权利要求书 3 页 说明书 16 页 附图 3 页

[54] 发明名称

一种磁盘阵列数据的读写方法及并行读写方法

[57] 摘要

本发明提供了一种磁盘阵列的数据读取方法、一种磁盘阵列的数据写方法、一种基于写数据过程中的数据读取方法、以及一种基于正常读数据过程中的数据写方法。本发明中，在读、写的过程中，使用分条单元为单元读取数据、或者写数据，以及在按照读写规则进行计算时使用分条单元为单位进行计算。并且，本发明所述基于写数据过程中的数据读取方法可以实现大写、小写、重建写或降级写的过程中的正常读或者降级读的并发，本发明所述基于正常读数据过程中的数据写方法，可以实现正常读过程中的大写、小写、重建写或降级写的并发。使用本发明，可以提高读写的速度，并尽量实现读写并发。



1、一种磁盘阵列 RAID 的数据读取方法，包括以下步骤：

A、根据当前的读规则，确定出要读取的数据所在扇区；

B、确定出要读取的扇区所属分条单元或校验数据单元，从磁盘读取确定出的各个分条单元和校验数据单元，存入缓存；

C、判断当前的读是否为正常读，若是，则执行步骤 D；否则，根据当前的读规则，使用缓存的分条单元和校验数据单元计算生成所要读取数据所属的分条单元，存于缓存；

D、从缓存中的分条单元读取出所需要的数据。

2、一种磁盘阵列 RAID 的数据写方法，包括以下步骤：

A、根据当前的写规则，确定出要预读的数据所在扇区；

B、确定出要预读扇区所属分条单元或校验数据单元，从磁盘读取确定出的各个分条单元及校验数据单元，存入缓存；

C、将要写入的数据与缓存中对应的分条单元进行处理，生成含有要写入数据的分条单元，作为要回写的分条单元，存于缓存；

D、根据当前的写规则，利用缓存的所读取出的分条单元和校验数据单元、生成的要回写的分条单元计算生成要回写的校验数据单元，存于缓存；

E、将所要回写的分条单元、要回写的校验数据单元回写入相应的磁盘中。

3、根据权利要求 2 所述的方法，其特征在于，步骤 A 所述根据当前的写规则，确定出要预读的数据所在扇区包括：

判断写为小写时，所述确定出的扇区包括：将要写入的数据所对应的扇区和对应的校验数据所在扇区；

判断写为大写时，所述确定出的扇区包括：将要写入数据所对应的扇区；

判断写为重建写时，所述确定出的扇区包括：重建写的写规则所要求预读的扇区、以及将要写入数据所对应的扇区；

判断写为降级写时，所述确定出的扇区包括：降级写的写规则所要求预读的扇区、以及校验数据所在扇区。

4、一种磁盘阵列 RAID 的基于写数据过程中的数据读取方法，其中，写数据包括将所要写入的数据所在分条单元读取到缓存中、运算生成含有要写入数据的分条单元和校验数据单元的预读，将生成的分条单元及校验数据单元写入磁盘的回写；所述数据读取方法包括以下步骤：

A、根据当前的读规则，判断缓存中是否存储有所有要读取数据所属分条单元，若是，则执行步骤 C；否则，执行步骤 B；

B、确定出未存于缓存中的要读取数据所属的分条单元；从磁盘中读取所述确定出的分条单元，存入缓存；

C、判断当前的读是否为正常读，若是，则执行步骤 D；否则，根据当前的读规则，使用缓存的分条单元和校验数据单元计算生成所要读取数据所属的分条单元，存于缓存；

D、从缓存的分条单元中取出所需要的数据。

5、根据权利要求 4 所述的方法，其特征在于，所述的写为大写、小写、重建写或降级写。

6、根据权利要求 4 所述的方法，其特征在于，步骤 D 所述的分条单元为写过程的预读出的分条单元，或者为预读后计算出的要回写的分条单元。

7、一种磁盘阵列 RAID 的基于正常读数据过程中的数据写方法，其中，正常读数据包括将所需数据所在分条单元读取到缓存中，从缓存的分条单元中取出/计算出所需数据提供给用户；所述数据写方法包括以下步骤：

A、判断缓存中是否存储有根据当前的写规则确定出的、要预读的数据所属的所有分条单元或/和校验数据单元，若是，则执行步骤 C；否则执行步骤 B；

B、确定出未存于缓存中的数据所属的分条单元或/和校验数据单元；从磁盘中读取所述确定出的分条单元或/和校验数据单元，存入缓存；

C、将要写入的数据与缓存中对应的分条单元进行处理，生成含有要写入数据的分条单元，作为要回写的分条单元，存于缓存；

然后根据所采用的写规则，利用缓存的所读取出的分条单元和校验数据单元、生成的要回写的分条单元计算生成要回写的校验数据单元；

D、将需要回写的分条单元、要回写的校验数据单元回写入相应的磁盘中。

8、根据权利要求 7 所述的方法，其特征在于，步骤 A 所述根据当前的写规则确定出的、要预读的数据所属的所有分条单元或/和校验数据单元包括：

判断写为小写时，所述分条单元包括：将要写入的数据所属分条单元、和校验数据单元；

判断写为大写时，所述分条单元包括：将要写入数据所属的分条单元；

判断写为重建写时，所述分条单元包括：重建写的写规则所要求预读数据所属分条单元、以及将要写入的数据所属分条单元；

判断写为降级写时，所述分条单元包括：降级写的写规则所要求预读的数据所述分条单元、以及校验数据单元。

9、根据权利要求 7 所述的方法，其特征在于，步骤 D 进一步包括：

将需要回写的分条单元回写入相应磁盘时，回写的分条单元恰为正常读数据操作当前正在读取的分条单元时，使用所述回写操作下发前缓存的该分条单元作为所述正常读数据操作当前读取的分条单元。

10、根据权利要求 9 所述的方法，其特征在于，所述的分条单元为写过程预读出的分条单元，或者为预读后计算出要回写的分条单元。

一种磁盘阵列数据的读写方法及并行读写方法

技术领域

本发明涉及磁盘阵列（RAID）的数据读写技术，特别是指一种磁盘阵列数据的读写方法及并行读写方法。

背景技术

磁盘阵列（RAID，Redundant Access Independent Disk）技术已作为一项成熟的技术应用于数据存储备份中。RAID5（RAID LEVEL 5）是磁盘阵列的一个标准。图 1 为一个由 4 个磁盘组成的 RAID5，4 个磁盘在逻辑上看作一个磁盘进行分条（Strip），本例中，分为分条 0～3。对于每一个分条，其中的 3 个磁盘用来存放数据，另外一个磁盘用来存放该分条的校验数据，每个分条分布在每个磁盘上的数据称为分条单元（SU，Strip Unit），分条单元也可称为分段（Block/Segment）。至于分条单元的大小视系统而定，有的系统或以 1KB 最有效率，或以 4KB，或以 6KB，甚至是 4MB 或 8MB。由于磁盘的读写是以扇区(Sector，512 字节)为单位，因此，分条单元应是 512 字节的倍数。如图 1 所示的 RAID5，分条 0 包括 SU0~SU2 和校验单元 P，每个分条单元包含 10 个磁盘扇区的数据。为了方便说明，本例中将分条 0 包含的各个磁盘的扇区与存储的数据编号为 0～29，存放校验数据的扇区与存储的校验数据编号为 P0～P9，将 P0～P9 称为校验数据单元。

下面仍以图 1 为例，给出了当前磁盘 0 故障的情况下，向扇区 9、10 写入数据的过程。由于磁盘 0 失效，该写请求可以视为拆分成如下的两个不同的“子写”请求，包括以下步骤：

1）对于写入新数据 9 的子写，包括：读扇区 19、读扇区 29，存于缓存中；由于要写的磁盘故障，因此按照降级写（Degraded Write）算法： P9=

原数据 19 XOR 原数据 29 XOR 新数据 9 ，生成新校验数据 P9 存于缓存中，其中 XOR 表示异或运算；将新的校验数据 P9 写入扇区 P9；

2）对于写入新数据 10 的子写，包括：读扇区 10、读扇区 P0，存于缓存中；按照小写（Small Write）或者称为读-修改-写（Read-Modify-Write）算法： P10 = 旧数据 10 XOR 新数据 10 XOR P0 ，生成新校验数据 P0，存于缓存中；将新数据 10 写入扇区 10，将新校验数据 P0 写入扇区 P0。

分析以上的过程，由于读写是针对扇区为单位进行，因此读写比较复杂，本例中，就包括了四次对磁盘读，两次计算，三次对磁盘写步骤。另一方面，由于以扇区为单位进行读写，若所读的扇区并不连续，即使是针对一个 SU 的正常读（Fault-Free Read）数据，例如读扇区 11、13，由于并不是针对连续扇区的读操作，也会分成两次读动作，频繁的对磁盘进行操作，影响数据的读写速度。

其中，上面提到的降级写算法和小写算法为 RAID5 中的写技术，写技术还包括大写算法（Large Write）和重建写（Reconstruct Write）其中小写、大写、重建写为正常写（Fault-Free Write），上面提到的正常读（Fault-Free Read）以及在后面提到的降级读（Degraded Read）算法为 RAID5 中的读技术，所述读写算法可以参见图 2 所示的读写规则算法示意图，其中图中 D 表示数据，P 表示校验数据。RAID 中的读写技术也可参见有关 RAID 的资料，如 CMU 大学（Carnegie Mellon University）1996 年 8 月出版的《RAIDFrame Book》或者该大学 1996 年 8 月出版的《RAIDFrame: A Rapid Prototyping Tool for RAID System》对 RAID 结构及涉及到的读写技术进行了详细的描述。

另外，不难理解当读、写命令针对同一块扇区时，可能会存在读写冲突。但是即使读、写命令并不是针对同一块扇区时，也可能会出现读写冲突。例如，对于 RAID5 处于降级（Degraded）状态，即 RAID 组存在一个磁盘失效的状态，在读取失效磁盘数据时，要通过读取其他磁盘的数据和校验数据，使用降级读算法计算出所需要的数据，因此即使存在的写命令并不是针对该

失效磁盘的扇区时，也可能存在着读写冲突，造成读数据不一致。下面详细分析这种情况。

仍以图 1 所示的 RAID5 为例，假设磁盘 0 坏掉而失效，假设当前存在着写分条 0 的扇区 11～18 的命令，同时存在读分条 0 的扇区 0～10 数据的命令。

数据写入扇区 11～18 的过程包括以下步骤：

步骤 1：根据 RAID5 小写（Small Write）算法，首先读取 SU1 上扇区 11～18 的旧数据 11～18_old，并从磁盘 3 读取出对应的旧校验数据 P1～P8_old，将读取的数据存于缓存（Cache）中。

步骤 2：将从磁盘 1 扇区 11～18 读取的旧数据及对应的旧校验数据与 Cache 中的要写入该扇区 11～18 的新数据按照小写算法：（P1～P8_new）=（11～18_old）XOR（11～18_new）XOR（P1～P8_old），生成新的校验数据（P1～P8_new）。由于步骤 1～2 的过程并没有对磁盘进行“写”动作，以上步骤 1～2 也合称为写动作中的预回写或预读（Preread）步骤。

步骤 3：将新数据 11～18_new 和新校验数据 P1～P8_new 分别回写入对应磁盘扇区，结束。

从磁盘 0 上的扇区 0～10 读数据过程包括以下步骤：

步骤 1：由于磁盘 0 坏，采用降级读算法，读取磁盘 1 上扇区 10～19 数据、磁盘 2 上扇区 20～29 的数据、校验数据 P0～P9，将读取的数据存于 Cache。

步骤 2：将步骤 1 读取的数据按照降级读算法：（0～9）=（10～19）XOR（20～29）XOR（P0～P9）得到磁盘 1 扇区 0～9 的数据，保存在 Cache 上。

步骤 3：将计算出来的数据 0～9 和在 Cache 中缓存的数据 10 作为要读取数据 1～10，提供给用户，结束。

分析上述读写步骤，读数据需要涉及读取不同磁盘 SU1 的 10～19、SU2

的 20～29、以及 P 的 P0～P9，由于是针对不同磁盘的连续扇区进行读取，因此相当于分成 3 个子读命令。而写操作要写 SU1 的 11～18 和 P 的 P1～P8，也可以认为是两个子写命令。当读写同时发生时，由于到达各个磁盘上的读写子命令无法保证顺序，就可能出现读写错误，例如当读数据完成对 10～19 的子读命令，而尚未执行读取 P0～P9 子命令时，若此时写操作完成了对 P1～P8 的子写命令，则读数据会读出更新后的校验数据。于是，便出现降级读数据时，从磁盘 1 读出的数据是旧数据，而从磁盘 3 读出的数据中却含有更新后的校验数据，导致在将读取的数据进行异或运算时，生成的数据 0～9 错误。

因此，为了避免读写操作的并行发生，目前对于读写的并行操作均进行加锁，并且加锁的单位为分条，这里的并行发生是指在一个操作的开始到完成之间出现另一个操作，例如写操作完成前出现读操作，或者在读操作完成前出现写操作。读写的加锁，即读数据时加锁禁止并行发生针对该分条的写操作；写数据时加锁禁止并行发生针对该分条的读操作。

通过以分条为单位进行加锁，不仅可以避免针对同一块扇区的读写冲突时，也可以避免上述针对不是同一块扇区的读写冲突。并且，通过以分条单位进行加锁，当写命令出现大写、小写、降级写，或者读命令出现正常读、降级读，不论是哪种写命令和读命令都可以避免同时出现，避免读写冲突，确保了数据的读写正确。

目前的加锁方法虽然保证了读写数据的正确性，但是，由于不论是哪种情况都进行加锁，使得对分条的读、写操作无法并行，影响读写操作的性能。会导致在写/读操作过程中下发命令后，用户必须等待当前操作进程结束解锁后，才可以对该分条进行读/写数据。因此加锁方式对读写并行操作时的性能影响较大，影响了读写速度。

发明内容

有鉴于此，本发明的主要目的在于提供一种磁盘阵列数据的读方法，以

减少读命令对磁盘的读指令的下发次数，提高数据读速度。

本发明的另一主要目的在于提供一种磁盘阵列数据的写方法，以减少写命令对磁盘的预读、回写指令的下发次数，提高数据写速度。

本发明的进一步的目的是提供一种磁盘阵列数据的基于写数据过程中的数据读取方法，以减少写数据中的对读操作的加锁情况，尽量实现读写的并行操作，提高数据读写速度。

本发明的进一步的目的是提供一种磁盘阵列数据的基于正常读数据过程中的数据写方法，以减少正常读数据过程中对写操作的加锁情况，尽量实现读写的并行操作，提高数据读写速度。

本发明提供了一种磁盘阵列的数据读取方法，包括以下步骤：

A、根据当前的读规则，确定出要读取的数据所在扇区；

B、确定出要读取的扇区所属的分条单元或校验数据单元，从磁盘读取确定出的各个分条单元和校验数据单元，存入缓存；

C、判断当前的读是否为正常读，若是，则执行步骤 D；否则，根据当前的读规则，使用缓存的分条单元和校验数据单元计算生成所要读取数据所属的分条单元，存于缓存；

D、从缓存中的分条单元读取出所需要的数据，提供给用户。

本发明还提供了一种磁盘阵列的数据写方法，包括以下步骤：

A、根据当前的写规则，确定出要预读的数据所在扇区；

B、确定出要预读扇区所属分条单元或校验数据单元，以分条单元为单位，从磁盘读取确定出的各个分条单元及校验数据单元，存入缓存；

C、将要写入的数据与缓存中的对应的分条单元进行处理，生成含有要写入数据的分条单元，作为要回写的分条单元，存于缓存；

D、根据当前的写规则，利用缓存的、所读取出的分条单元、生成的要回写的分条单元和校验数据单元计算出要回写的校验数据单元，存于缓存；

E、以分条单元为单位，将所要回写的分条单元、要回写的校验数据单

元回写入相应的磁盘中。

其中该写方法中，步骤 A 所述根据当前的写规则，确定出要预读的数据所在扇区包括：

判断写为小写时，所述确定出的扇区包括：将要写入的数据所对应的扇区和对应的校验数据所在扇区；

判断写为大写时，所述确定出的扇区包括：将要写入数据所对应的扇区；

判断写为重建写时，所述确定出的扇区包括：重建写的写规则所要求预读的扇区、以及将要写入数据所对应的扇区；

判断写为降级写时，所述确定出的扇区包括：降级写的写规则所要求预读的扇区、以及校验数据所在扇区。

本发明还相应提供了一种基于写数据过程中的数据读取方法，其中，写数据包括将所要写入的数据所在分条单元读取到缓存中、运算生成含有要写入数据的分条单元的预读，将生成的分条单元及校验数据单元写入磁盘的回写；所述数据读取方法包括以下步骤：

A、根据当前的读规则，判断缓存中是否存储有所有要读取数据所属分条单元，是，则执行步骤 C；否则，执行步骤 B；

B、确定出未存于缓存中的要读取数据所属的分条单元；从磁盘中读取所述确定出的分条单元，存入缓存；

C、判断当前的读是否为正常读，若是，则执行步骤 D；否则，根据当前的读规则，使用缓存的分条单元和校验数据单元计算生成所要读取数据所属的分条单元，存于缓存；

D、从缓存的分条单元中取出所需要的数据，提供给用户。

其中，所述的写为大写、小写、重建写或降级写。

其中该写数据过程中的读方法中，步骤 D 所述的分条单元为写过程的预读出的分条单元，或者为预读后计算出的要回写的分条单元。

本发明还相应提供了一种基于正常读数据过程中的数据写方法，其中，

正常读数据包括将所需数据所在分条单元读取到缓存中，从缓存的分条单元中取出/计算出所需数据提供给用户；所述数据写方法包括以下步骤：

A、判断缓存中是否存储有根据当前的写规则确定出的、要预读的数据所属的所有分条单元或/和校验数据单元，若是，则执行步骤 C；否则执行步骤 B；

B、确定出未存于缓存中的数据所属的分条单元或/和校验数据单元；从磁盘中读取所述确定出的分条单元或/和校验数据单元，存入缓存；

C、将要写入的数据与缓存中对应的分条单元进行处理，生成含有要写入数据的分条单元，作为要回写的分条单元，存于缓存；

然后根据所采用的写规则，利用缓存的所读取出的分条单元和校验数据单元、生成的要回写的分条单元计算生成要回写的校验数据单元；

D、将需要回写的分条单元、要回写的校验数据单元回写入相应的磁盘中。

其中，该读数据过程中数据写方法中，步骤 A 所述根据当前的写规则确定出的、要预读的数据所属的所有分条单元或/和校验数据单元包括：

判断写为小写时，所述分条单元包括：将要写入的数据所属分条单元、和校验数据单元；

判断写为大写时，所述分条单元包括：将要写入数据所属的分条单元；

判断写为重建写时，所述分条单元包括：重建写的写规则所要求预读数据所属分条单元、以及将要写入的数据所属分条单元；

判断写为降级写时，所述分条单元包括：降级写的写规则所要求预读的数据所述分条单元、以及校验数据单元。

其中，该读数据过程中数据写方法中，步骤 D 进一步包括：将需要回写的分条单元回写入相应磁盘时，回写的分条单元恰为正常读数据操作当前正在读取的分条单元时，使用所述回写操作下发前缓存的该分条单元作为所述正常读数据操作当前读取的分条单元。其中，所述的分条单元为写过程的

预读出的分条单元，或者为预读后计算出要回写的分条单元。

由上述方法可以看出，本发明针对分条单元进行读、写操作时，并不是针对各个扇区，因此针对一个分条单元的多个不连续的扇区的读写可以一次读写，降低了对磁盘的读写次数。另一方面，计算也是以分条单元为单位进行，可以将针对一个分条单元的不连续的扇区的运算简化成一个运算过程，减少运算次数，提高了读写速度。

另一方面，相对于 RAID 现有技术的读写并行时无论何种情况均加锁来说，采用本发明提供的基于写数据过程中的数据读取方法和基于正常读数据过程中的数据写方法，仅仅在降级读过程中出现回写时需要进行分条加锁来禁止回写，其他情况下的读写都可以并行发生，不需要进行加锁。也就是说，使用本发明，在读写数据的并行过程中，当写数据过程中出现正常读或降级读；或者正常读数据过程中出现写数据，均无需加锁，这里的写包括大写、小写、降级写、重建写。

由于可以尽量的实现不加锁，使得读/写数据请求尽量不受写/读数据的命令影响，加快了读写并行时的数据的读写速度，提高了读写性能。

附图说明

图 1 为 RAID5 示意图。

图 2 为 RAID5 中的读写规则算法示意图。

图 3 为本发明读数据流程。

图 4 为本发明写数据流程。

图 5 为本发明写数据中的读数据流程。

图 6 为本发明读数据中的写数据流程。

具体实施方式

本发明采用了以分条单元为单位进行读写操作，而不是以扇区为单位进

行读写操作，也就是说，在对某扇区进行读写操作时，会读写该扇区所在的整个分条单元。基于分条单元为单位的读写方法，可以简化读写过程，提高读写速度。下面通过具体实施例和附图，对本发明进一步详细说明。

本发明提供的读数据方法，可应用于 RADI5 的正常读、降级读，参见图 3 所示的流程图，包括以下步骤：

步骤 301：根据当前的读规则，确定要读取的扇区。也就是根据是正常读还是降级读来确定所要读取的扇区。如图 2，正常读，确定的扇区位于 D0；若为降级读，确定出的扇区位于 D1、D2、P。这个步骤和现有的技术一样。

步骤 302：确定出要读取的数据所属的分条单元，以分条单元为单位，读取确定出的各个分条单元或/和校验数据单元存入缓存。如图 2，为正常读时，仅读取数据所属的分条单元 D0 存入缓存；当为降级读时，要读取分条单元 D1、D2、校验数据单元 P 存入缓存。

步骤 303：根据读规则，计算出所要读取数据所在分条单元。当正常读时，其实这个步骤为空，跳过直接执行步骤 D；当为降级读时，根据降级读的规则，利用读取出的分条单元和校验数据单元计算出所要读的数据所在的分条单元，存于缓存。其中，计算的算法规则与图 2 中示出的相同，只是这里以分条单元为单位进行计算。

步骤 304：从缓存中的分条单元读取出所需要的扇区数据，提供给用户。这里所述的缓存中的分条单元，包括步骤 B 和 C 所缓存的分条单元。

本发明提供的 RAID 写数据方法，可应用于大写、小写、降级写、重建写，参见图 4 示出的流程图，包括以下步骤：

步骤 401：根据当前的写规则，确定出要预读（Preread）的扇区，也就是根据是大写、小写、降级写、重建写来确定要读取的扇区。对于小写，确定出的扇区为将要回写的数据所在扇区和对应的校验数据的扇区；这个步骤和现有的技术一样。而对于大写和重建写，本发明确定出的扇区不仅包括要

预读的扇区，也要包括将要回写的数据所在扇区。对于降级写，本发明确定的扇区包括要回写的数据所在扇区和校验数据所在扇区。

如图 2，对于小写，确定出的扇区位于 D0、P；对于大写，确定出的扇区位于 D0、D1、D2；对于重建写，确定出的扇区位于 D0、D1、D2；对于降级写，确定出的扇区位于 D1、D2、P。

步骤 402：确定出要预读扇区所属分条单元或校验数据单元，以分条单元为单位，读取确定出的各个分条单元及校验数据单元，存入缓存。

步骤 403：将要写入的数据与缓存中的分条单元进行处理，生成含有要写入数据的分条单元，作为要回写的分条单元，存于缓存。由于这个处理过程可以看作是将要写入的数据补满成一个分条单元，因此这个过程本发明中称为补满过程。如图 2，也正是因为存在这个补满过程，因此，本发明中在步骤 401～402，对于大写，要将 D0、D1、D2 分条单元读取到缓存；对于重建写，要将 D0、D1 分条单元读取到缓存；以及对于降级写，要一同读取校验数据单元 P，以使用降级读算法计算出分条单元 D0，相当于读取 D0。

步骤 404：根据当前的写规则，利用缓存的读取出的分条单元、补满后的要回写的分条单元和/或校验数据单元计算出要回写的校验数据单元，存于缓存。其中，计算的算法规则与图 2 中示出的相同，只是这里以分条单元为单位进行计算。例如图 2 中的小写，则要利用步骤 402 读出缓存的分条单元 D0、P、补满后要回写的分条单元 D0 来计算出要回写的校验数据单元 P。

步骤 405：以分条单元为单位，将所要回写的分条单元、校验数据单元回写入相应的磁盘中。

下面结合具体的实施例来分析采用分条单元为读写单位的优点。仍以背景技术中的例子对本发明的读写进行说明：当前磁盘 0 故障的情况下，写入数据 9、10 的过程。为了与背景技术所述的过程进行对比，下面的步骤没有与图 4 示出的写流程进行对照写，但是，其处理流程仍然是依照图 4 示出的步骤。包括以下步骤：

步 1：判断9、10所在分条单元，读取分条单元10～19、20～29，校验数据单元P0～P9；异或生成旧数据0～9，存于缓存；

步 2：利用缓存中的分条单元数据，以及要写入的数据，补满SU0、SU1在Cache中的要回写的分条单元。本例中，补满后的SU0为：0～8是旧数据，9是要写入的新数据；补满后的SU1为：10是要写入的新数据，11～19为旧数据。

步 3：按照大写方式，利用Cache中补满后的分条单元SU0、SU1，以及SU2，异或生成新校验数据单元P；然后回写补满的分条单元SU1、回写生成的新校验数据单元P到磁盘。

以上可以分析出，该写过程包括：三次对磁盘读，两次异或运算，两次对磁盘写。相比背景技术中的做法，少了一次对磁盘读、一次对磁盘写简化了处理流程，提高了写性能。

尤其是相对于背景技术中提到的对一个 SU 的不相邻的扇区进行读/写时，由于扇区不连续，背景技术中会分开两次进行读/写，但是使用本发明，由于读是针对分条单元，而写采用了补满技术，因而写也是针对分条单元，均是针对连续扇区进行 作，因此读/写也会看作一次读/写处理，大大减小了对磁盘的读写次数，简化了处理步 ，提高了读写的性能。

由于采用了以分条单位进行读写，降低了读写的复杂性，因此，本发明提供了读写并行的可能性。下面进行分析：

首先分析RAID5正常读与写过程。如图1所示，假设需要写扇区8、11～18、P1；要读取 10、19、20～29。在读写并行发生时，可能存在以下情况：

情况1：正常读时，预读启动；或预读时，正常读又下发：

预读步 为：

①、根据要写的数据，判断出并读取数据所在的分条单元SU0、SU1、校验数据单元P，存于Cache；

使用补满技术处理缓存中的分条单元，生成要回写的补满后的分条单元，存于 Cache。本例中，用缓存的 0～7 和 9 补满要写的数据 8 生成要回写的 SU0，用缓存的 10 和 19 补满要回写的数据 11～18 生成要回写的 SU1；

②、根据小写算法，利用补满后的 SU0、补满后的 SU1、旧的校验数据单元 P 进行异或运算生成新的校验数据单元 P，存于 Cache。

不难理解，由于预读仅有读请求下发，与正常读不会出现任何冲突。

情况 2：正常读完成前，回写启动，也就是在读过程中出现回写：

在本例中，所读取和回写的扇区并不存在交集，因此实际上，虽然回写过程中会将还没有读取的分条单元进行更新，造成读出的一些分条单元是旧分条单元，一些是新分条单元，但由于最终提供给用户的数据对应的扇区数据没有变化，因此，回写造成影响也只是过程中的影响，而对于最终的读出给用户的结果是没有影响的，因此，当如本例所读取和回写的扇区并不存在交集的情况，可以不对写进行加锁，允许该写进行。

但是，若所读取和回写的扇区存在交集的话，例如，回写的数据还包括 10，则当读 SU1 的时候，若发现针对该单元的回写存在，则表示预读已经完成，可使用启动回写动作前所生成的、保存在缓存中的 SU1 来取代要读取的 SU1。这样，读取提供给用户的数据将是最新的数据，也就是更新的要回写的 10，因此这种情况下不会出现读写冲突，可以不对写进行加锁，允许该写进行；当然，也可以采用启动回写前所预读的 SU1 来取代要正常读的 SU1，只是最终提供给用户数据是写操作启动前的数据。

情况 3：回写完成前，正常读又下发，也就是在写的过程中，出现的读：

由于回写开始，表示预读已结束，根据预读的步骤，10～19 的数据已在 Cache 中，可直接从 Cache 读出数据；数据 20～29 Cache 中没有，则下发请求到磁盘读取 SU2，将数据读取到缓存后，再提取出数据发送给用户。这种情况下不会出现读写冲突，可以不对写进行加锁，允许该写进行。并且当采用的是预读时计算后的 SU1 时，最终读取提供给用户的数据是最新数

据。

下面再分析RAID5降级读与写并行出现的过程。

仍以图1为例，假设磁盘0坏，要读数据0～10，要写数据11～18。在读写同时发生时，可能存在以下情况：

情况1：降级读时，预读又启动；或预读时，降级读又下发：

降级读步 为：

①：根据要读的数据，判断并读取分条单元SU1、SU2、校验数据单元P，存于Cache；

根据降级读算法，异或生成SU0，存于Cache，从缓存中将数据0~10提供给用户。

预读步 为：

①：根据要写的数据，判断出并读取数据所在的分条单元SU1；读取校验数据单元P，存于Cache；

使用补满技术处理缓存中的分条单元，生成要回写的补满后的分条单元，本例中，用缓存的10和19补满要写的数据11～18生成要回写的SU1，存于Cache；

②：根据小写算法，利用新的SU1、旧的P生成新的校验数据单元P，存于Cache。

不难理解，降级读和预读是两个读命令，因此不会产生冲突，不需要进行加锁，即只要读完成前，回写没有开始，非命中读和预读间就不可能冲突。

情况2：降级读完成前，回写启动，即降级读过程中，出现回写：

由于回写将修改P1，而降级读又要读取P1旧数据，因此降级读和回写间存在冲突，需要 加“分条锁”以确保数据的一致性。即在降级读开始时，降级读会对该分条加“分条锁”禁止回写，回写命令将暂停执行，以确保数据一致性。

情况3：回写开始后，降级读又下发，即在回写的过程中出现降级读：

根据回写算法，在回写开始时，已经完成预读步骤，SU1及校验数据单元P经预读步骤必定已存于Cache中，因此，降级读可以从Cache中读取到新SU1、新校验数据单元P，而通过读取磁盘获得SU2，异或生成SU0。不难理解，降级读也可以从Cache中读取到旧SU1、旧校验数据单元P，而通过读取磁盘获得SU2，同样异或生成SU0。而后从缓存中将生成的0～9及10提供给用户。可以看出，这种情况下也可以不用加锁，实现降级读。

本例中的情况3是以小写为例，在降级读的情况下也可能出现降级写，从本发明图4对应的写方法来看，降级写的预读过程在进行补满操作前，必然首先生成故障磁盘的分条单元，因此，在进行降级写的回写启动时，缓存中必然已经存在着一个分条的所有分条单元和校验单元，因此若出现任何读，都可以从缓存中找到所要读取的数据，因此可以实现这种情况下的写。

通过以上的分析，使用基于分条单元的读写方法，不需要和现有的RAIDFrame一样，对于任何情况都要加锁，仅在降低读过程中，出现的回写才需要加锁。其余情况，读写间不存在任何互斥，都不需要进行加锁，可以实现读写的并行、缩短的等待时延，因此提高了数据读写速度，提高了性能。

基于以上的分析，在本发明所述的RAID的写方法、读方法的基础上，本发明又提供了在写的过程中的读的方法，和在读的过程中的写的方法，以实现所述的RAID尽量不需要加锁的读写并行过程的方法，如下：

本发明提供的基于写数据的过程中的读数据方法，应用于所述的读为正常读（Fault-Free Read），或者降级读（Degraded Read），写可以为任意写的情况。其中，写过程包括预读和回写过程，所述写过程中的读包括以下步骤：

步骤501：根据读规则，首先判断缓存中是否存储有所有要读数据所在分条单元，是，则对缓存的分条单元进行操作，执行步骤503；否则，执行步骤502。以图2为例说明，当为正常读时，则判断缓存中是否存在分条单

元 D0；当是降级读时，首先判断是否存在分条 D0，若无，则进一步判断是否存在 D1、D2 和 P。

步骤 502：确定出未保存在缓存中的、要读取的数据所在的分条单元，非命中读取数据所在分条单元，存入缓存。这里说的非命中读是指直接从磁盘中读取数据，对应的命中读则指直接从缓存中读取数据。

步骤 503：根据当前的读规则，从缓存的分条单元中取出或计算出所需要的数据，提供给用户。如图 4 所示，对于正常读来说，直接从缓存的分条单元 D0 中读取出所需数据提供给用户即可；对于降级读来说，则需要将缓存中的 D1、D2 和 P 进行异或运算出分条单元 D0，然后再从缓存的分条单元 D0 中读取出所需数据提供给用户。

本发明还提供了基于正常读（Fault-Free Read）数据过程中的写数据方法，应用所有写的情况，其中，正常读数据包括将所需数据所在分条单元读取到缓存中，从缓存的分条单元中取出所需数据提供给用户；所述正常读数据过程中的写包括以下步骤：

步骤 601：在写规则的预读（Preread）过程，判断缓存中是否存有所有需要预读的数据所在的分条单元，是，则执行步骤 603；否则步骤 602；这里的写规则包括大写、小写、降级写的规则。其中，在确定哪些数据是需要读取的时候，可以参见步骤 401。

步骤 602：确定出未保存在缓存中的、需要预读取数据所在的分条单元；然后非命中读取所述确定出的数据所在分条单元，存入缓存；

步骤 603－604：将要回写的数据与缓存中的分条单元进行处理，生成含有要回写数据的、补满后的分条单元，作为要回写的分条单元，存于缓存，然后根据所采用的写规则，以分条单元为单位进行运算，计算出要回写的校验数据单元。其中，计算校验单元的规则可以参考图 2 所示，只不过本发明是以分条单元为单元进行计算。

步骤 605：将计算出的需要回写的分条单元、校验数据单元回写入相应

的磁盘中。若当回写时，回写的分条单元恰为正常读数据当前正在读取的分条单元时，则继续回写，而对于这个正常读，则使用该写操作时所缓存的分条或者是补满计算后的分条单元作为要读取的分条单元即可，也就是说，当用磁盘读出的数据存入缓存时，若与写操作缓存中的数据存在交集部分，则须以写操作过程中已经缓存在缓存中的数据为准。

通过以上可以看出，仅仅在降级读过程中出现回写时需要分条锁禁止写，其他情况下的读写都可以使用上面的步骤进行，不需要进行加锁。

由于 RAID0（RAID LEVEL 0）、RAID1（RAID LEVEL 1）没有校验单元，仅存在对磁盘的直接读、直接写的步骤，没有使用校验的计算过程（具体可参见背景技术中提到的 CMU 大学出版的两本书），因此本发明所述的以分条单元为单位进行读写，也适用于 RAID0、RAID1，其读方法和写方法和本发明所述方法相似，不同点在于将所述的按照规则计算的过程省去。并且依照本发明读方法、写方法，就可以实现 RAID0、RAID1 所读写并行的情况，读写均无需加锁。

以上所述仅为本发明的较佳实施例而已，并不用以限制本发明，凡在本发明的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。

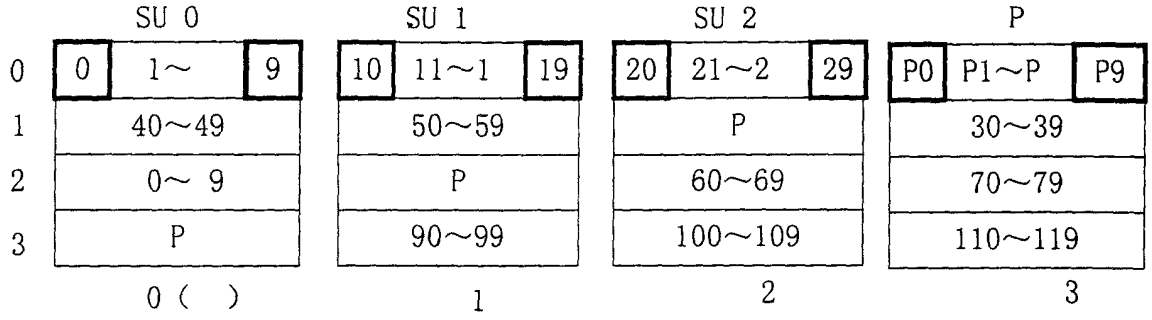
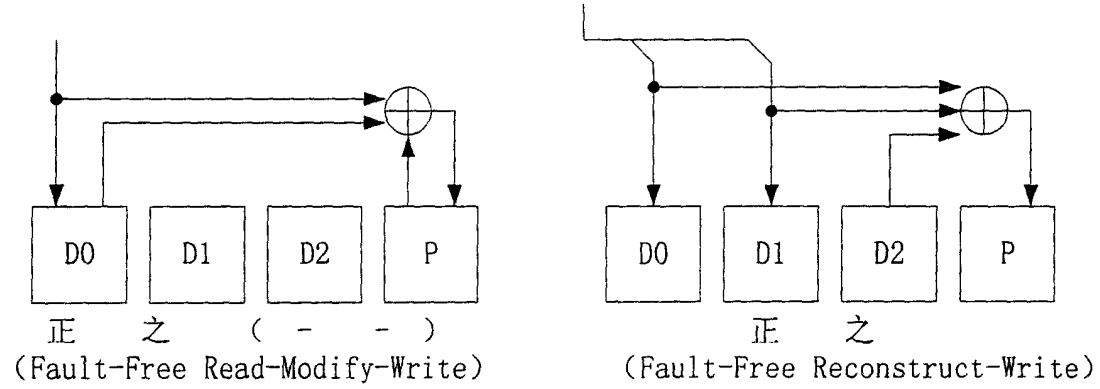
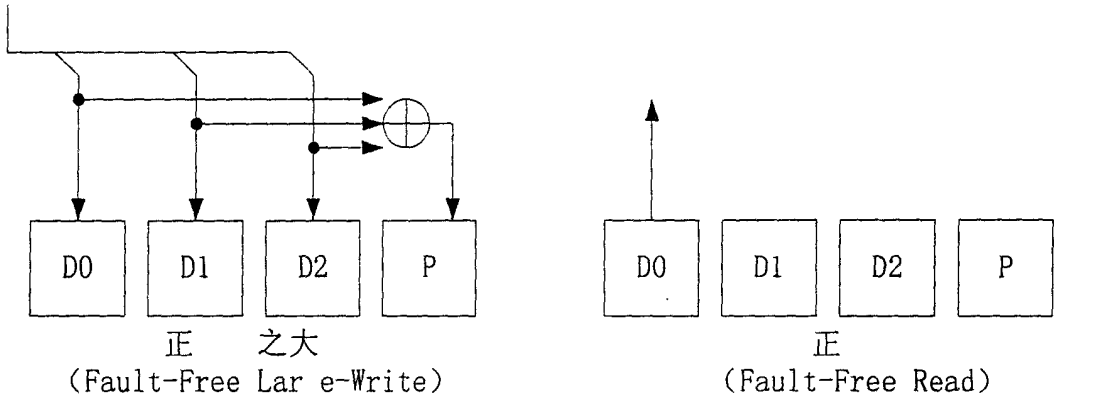

SU 0
SU 1
SU 2
P
0
1
2
3
0 1～ 9
10 11～1 19
20 21～2 29
P0 P1～P P9
40～49
0～ 9
P
50～59
P
90～99
60～69
100～109
30～39
70～79
110～119
0( )
1
2
3

图 1


D0
D1
D2
P
正 之 (- -)
(Fault-Free Read-Modify-Write)
正 之
(Fault-Free Reconstruct-Write)


D0
D1
D2
P
正 之大
(Fault-Free Lar e-Write)
正
(Fault-Free Read)

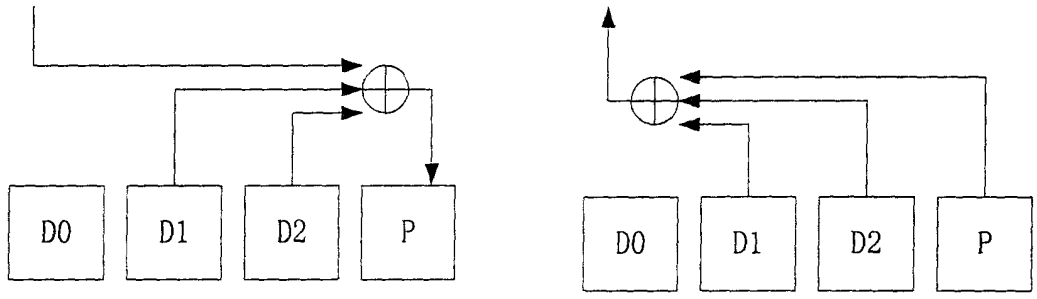

D0
D1
D2
P
(De raded Write)
(De radedRead)

图 2

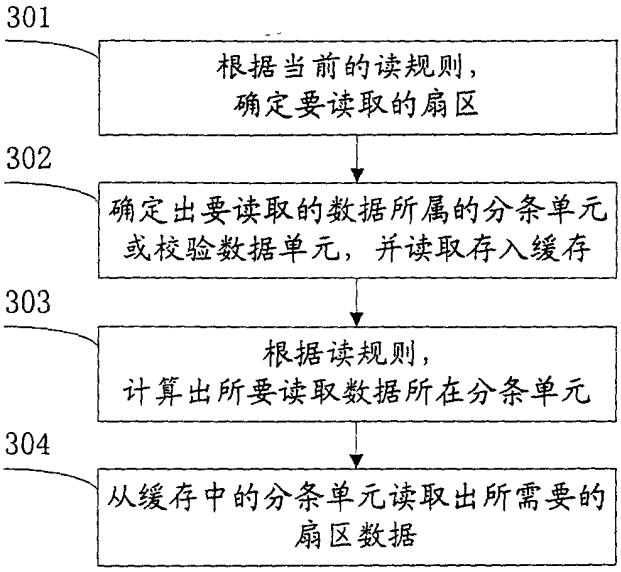

301
根据当前的读规则，
确定要读取的扇区
302
确定出要读取的数据所属的分条单元
或校验数据单元，并读取存入缓存
303
根据读规则，
计算出所要读取数据所在分条单元
304
从缓存中的分条单元读取出所需要的
扇区数据

图 3

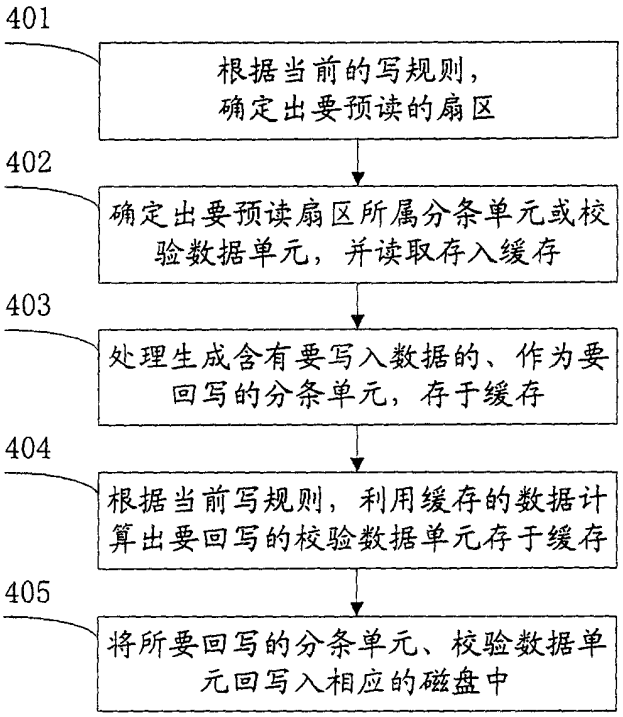

401
根据当前的写规则，
确定出要预读的扇区
402
确定出要预读扇区所属分条单元或校
验数据单元，并读取存入缓存
403
处理生成含有要写入数据的、作为要
回写的分条单元，存于缓存
404
根据当前写规则，利用缓存的数据计
算出要回写的校验数据单元存于缓存
405
将所要回写的分条单元、校验数据单
元回写入相应的磁盘中

图 4

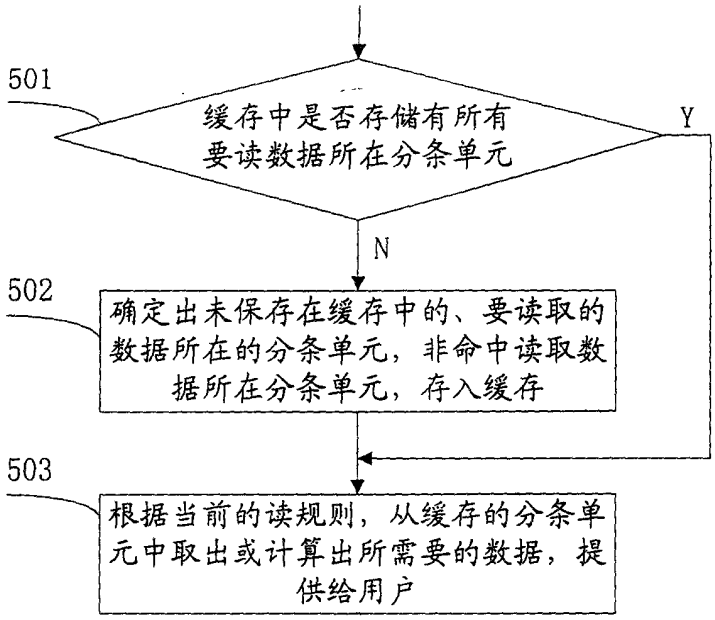

501
缓存中是否存储有所有要读数据所在分条单元
Y
N
502
确定出未保存在缓存中的、要读取的数据所在的分条单元，非命中读取数据所在分条单元，存入缓存
503
根据当前的读规则，从缓存的分条单元中取出或计算出所需要的数据，提供给用户

图 5

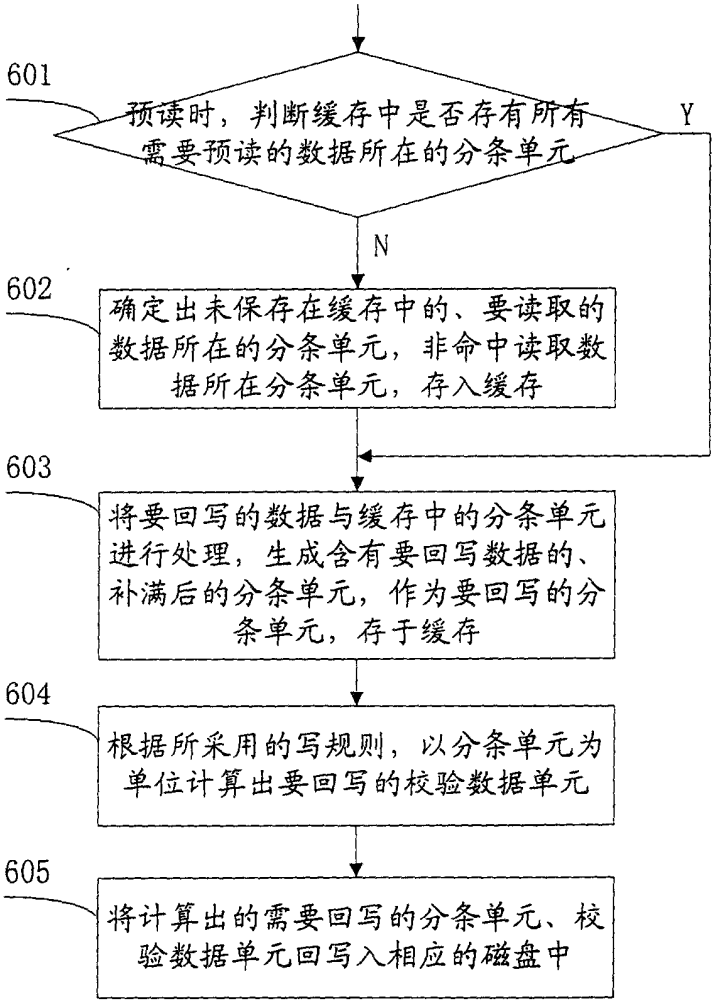

601
预读时，判断缓存中是否存有所有需要预读的数据所在的分条单元
Y
N
602
确定出未保存在缓存中的、要读取的数据所在的分条单元，非命中读取数据所在分条单元，存入缓存
603
将要回写的数据与缓存中的分条单元进行处理，生成含有要回写数据的、补满后的分条单元，作为要回写的分条单元，存于缓存
604
根据所采用的写规则，以分条单元为单位计算出要回写的校验数据单元
605
将计算出的需要回写的分条单元、校验数据单元回写入相应的磁盘中

图 6