(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101339808 B
(45)授权公告日 2011.02.09

(21)申请号 200810048591.4

(22)申请日 2008.07.28

(73)专利权人 华中科技大学
地址 430074 湖北省武汉市洪山区珞瑜路1037号

(72)发明人 余鑫 吴小龙

(74)专利代理机构 北京市德权律师事务所 11302
代理人 张伟

(51)Int.Cl.
G11C 16/10(2006.01)
G06F 12/02(2006.01)

(56)对比文件
US 2008/0162786 A1,2008.07.03, 全文.
US 2007/0030734 A1,2007.02.08, 全文.
US 2002/0116569 A1,2002.08.22, 全文.
CN 1815629 A,2006.08.09, 全文.

审查员 李元

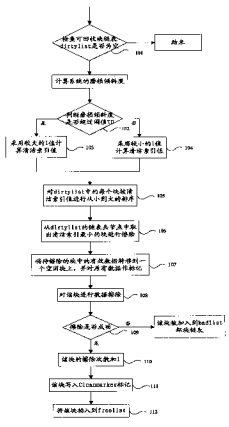
权利要求书 2 页 说明书 6 页 附图 2 页

(54)发明名称

存储块的擦除方法及装置

(57)摘要

本发明公开了一种存储块的擦除方法，包括：根据含有无效数据的存储块的磨损均匀程度，动态调整回收效率与磨损平衡间的倾向，获取所述存储块的清洁索引值，根据所述清洁索引值确定要擦除的存储块进行数据擦除。本发明还公开了一种存储块的擦除装置，包括动态调整模块、清洁索引模块和擦除模块。本发明通过设置一种新的清洁索引计算方法，引入了新的参量，使得当区块擦除次数接近擦除极限时，仍然能有效计算块间的磨损平衡。同时，还提出了一种动态的调整方法，根据块的磨损均匀程度，动态调整回收效率与磨损平衡之间的倾向，保证闪存的垃圾回收中有较平衡的性能以及自适应能力。



CN 101339808 B

1. 一种存储块的擦除方法，其特征在于，包括以下步骤：

根据含有无效数据的存储块的磨损均匀程度，动态调整回收效率与磨损平衡间的倾向，获取所述存储块的清洁索引值，根据所述清洁索引值确定要擦除的存储块进行数据擦除；

所述获取清洁索引值具体包括：

采用如下公式获取各个存储块的清洁索引值：

$$cleaningIndex = (1-l)u_i + l \times \frac{\varepsilon_i - \varepsilon_{\min}}{\varepsilon_{\max} - \varepsilon_{\min} + 1},$$

其中，u_i 表示当前存储块的利用率、ε_i 表示当前存储块擦除次数、ε_{min} 和 ε_{max} 表示所有存储块的最小擦除次数和最大擦除次数，1 为正规下齐平度。

2. 根据权利要求 1 所述的方法，其特征在于，该方法进一步包括：

建立含有无效数据的存储块的块信息的数据结构，在该数据结构里面保存有该存储块的清洁索引的值，并采用链表的方式存储，该链表根据清洁索引的值进行排序；

建立不含无效数据的非空闲数据的块信息的数据结构，在该数据结构里面保存有该存储块的空闲空间及当前存储块擦除次数。

3. 根据权利要求 2 所述的方法，其特征在于，所述动态调整回收效率与磨损平衡间的倾向具体包括：

获得磨损倾斜度的值，所述磨损倾斜度的获取方法为：

$\Delta_\varepsilon = \varepsilon_{max} - \varepsilon_{min}$，

判断该磨损倾斜度的值是否超过一阈值，并调整清洁索引值的计算参数 1。

4. 根据权利要求 3 所述的方法，其特征在于，所述根据清洁索引值确定要擦除的存储块进行数据擦除操作具体包括：

对含有无效数据存储块的链表中的每个存储块根据其清洁索引值进行从小到大的排序；

在所述链表的头节点中取出清洁索引最小的存储块信息；

将所述存储块中的有效数据转移到一个空闲块上，并对所述存储块的原有数据作标记，标记为无效数据；

对所述存储块进行数据擦除，以回收作为空闲块使用。

5. 根据权利要求 4 所述的方法，其特征在于，该方法进一步包括：

判断对所述存储块的擦除操作是否成功，如果是，则将该存储块的当前存储块擦除次数值加 1，并写入清洁标记，加入空闲块链表；否则将所述存储块加入坏存储块链表。

6. 根据权利要求 3 所述的方法，其特征在于，

所述计算参数的调整方式为：判断磨损倾斜度是否超过一阈值，如果是，则采用较大的正规下齐平度，使得当前存储块擦除次数值在清洁索引值的计算中占较大比重；否则采用较小的正规下齐平度，使得块的利用率在清洁索引值的计算中占较大比重；

当要进行更精确的控制时，对当前存储块擦除次数进行分段，当所述当前存储块擦除次数值低于另一阈值时，也使用较大的正规下齐平度。

7. 一种存储块的擦除装置，其特征在于，包括：

动态调整模块，用于根据含有无效数据的存储块的磨损均匀程度，动态调整回收效率

与磨损平衡间的倾向；

清洁索引模块，用于采用如下公式获取各个存储块的清洁索引值：

$$cleaningIndex = (1-l)u_i + l \times \frac{\varepsilon_i - \varepsilon_{\min}}{\varepsilon_{\max} - \varepsilon_{\min} + 1},$$

其中，u_i 表示当前存储块的利用率、ε_i 表示当前存储块擦除次数、ε_{min} 和 ε_{max} 表示所有存储块的最小擦除次数和最大擦除次数，l 为正规下齐平度，来获取所述存储块的清洁索引值；

擦除模块，用于根据所述清洁索引值确定要擦除的存储块并进行数据擦除。

8. 根据权利要求 7 所述的装置，其特征在于，所述动态调整模块具体包括：

判决单元，用于获得磨损倾斜度的值，判断该值是否超过一阈值；

选择单元，用于根据所述判决单元的判断结果，如果该值超过所述阈值，则采用较大的正规下齐平度；否则采用较小的正规下齐平度。

9. 根据权利要求 7 或 8 所述的装置，其特征在于，所述擦除模块具体包括：

排序单元，用于对含有无效数据存储块的链表中的每个存储块根据其清洁索引值进行从小到大的排序；

定位单元，用于在所述链表的头节点中取出清洁索引最小的存储块信息；

数据迁移单元，用于将所述存储块中的有效数据转移到一个空闲块上，并对所述存储块的原有数据作标记，标记为无效数据；

数据擦除单元，用于对所述存储块进行数据擦除。

存储块的擦除方法及装置

技术领域

[0001] 本发明涉及闪存的管理技术，特别是存储块擦除方法及装置。

背景技术

[0002] Flash 存储卡（闪存）主要应用于智能电话、数码相机、PDA 等相关领域，由 Flash 存储芯片和存储卡控制电路这两部分的组成。其中，Flash 存储芯片是 Flash 存储卡的存储实体，它是一种基于半导体的存储器，具有功耗低、容量大、访问速度高、无机械故障，以及数据非易失的优点。随着 Flash 存储芯片容量的飞速增长，人们对数据操作的灵活性提出了越来越高的要求，对 Flash 存储芯片中的数据存储管理已成为一个不容回避的问题。

[0003] 在闪存经过反复读写之后，块内与块间会有大量的文件碎片，这时需要进行垃圾收集。因为，在很多块中，含有脏数据－即无效数据－的块需要将脏数据占据的存储空间回收，这时就需要将含有脏数据的块回收，将块中的有效数据重写到其他空闲块中，并擦除该块。这个过程称为垃圾收集。在垃圾收集过程时，首先要重写块的全部有效数据，然后再擦除整个块。尽管 Flash 存储芯片作为非易失性存储器，可以反复地编程并擦除，但是在其每个存储块被磨损坏之前，只能对其进行一定次数的擦除。在某些系统中，在存储块被认定为不可用之前，可对其进行约一万次的擦除。当存储块被磨坏时，会进而导致整个闪存存储容量的一部分无法使用或其性能的重大下降，导致丢失已存储数据或无法存储数据，使用户受到不良影响。

[0004] 每个闪存存储块可以根据其擦除次数分为年老和年轻的区块，擦除次数越多年龄越老，反之则越年轻。在闪存文件系统中，需要考虑对闪存芯片的磨损平衡，其目的就是将整个闪存中的各个块的年龄差异控制在一定的范围，使整个闪存磨损程度趋于一致，延长闪存的寿命。如果总是擦除年老的块，那么会导致年老块的加速损坏，影响整个闪存的使用。因此在垃圾收集擦除闪存块时，应该尽量回收年轻的区块来擦除。

[0005] 块的利用率是指存储块上有效数据所占的比例。垃圾收集首先把待擦除块的有效数据写到当前块中，再对那个待擦除块进行擦除。其中擦除操作时间是固定的，重写数据耗费的时间与重新写入的数据量有关，并决定着垃圾收集时间的长短。随着区块利用率的提高，每次重写的数据量越大，那么重写所耗费的时间也越长，回收的效率也越低。据资料显示，当闪存块的利用率达到 80％以上时，重写的代价会迅速提高。因此在垃圾收集中，选择擦除块时，考虑磨损平衡的同时，还要考虑到回收效率的问题。应该尽量选择利用率低，且年龄较小的块。

[0006] 一种典型的磨损平衡算法是贪婪算法，它根据各旧数据块被标记的时间，每次回收最先被标记的旧数据块，直到所有的旧数据区块都被成功擦除回收。这种算法资源占用率低，避免了回收操作时无序的回收旧数据块，但是它没有考虑年龄因素，有可能每次回收最先标记的旧数据块，而该旧数据块恰好是年龄较大的块，这就会使得对该块的磨损加剧。

[0007] 在垃圾收集过程中，需要考虑块的年龄和块的利用率两个因素。为了修正贪婪算法的不足，Kim and Lee 重新定义了选择擦除块的指标，定义了清洁索引的概念，弥补了贪

婪算法的不足。清洁索引的定义如公式 1 所示，其中 u_i 表示块 i 的利用率、ε_i 表示当前擦除次数、ε_{min} 和 ε_{max} 表示块的最小擦除次数和最大擦除次数。

[0008] $$cleaningIndex = (1-l)u_i + l \times \frac{\varepsilon_i}{\varepsilon_{max}+1} \quad \text{（公式 1）}$$

[0009] 公式 1 中的 l 为正规下齐平度，其计算方法使用下面的公式 2，其中 $\Delta\varepsilon = \varepsilon_{max} - \varepsilon_{min}$ 表示磨损倾斜度，反应了磨损的平衡程度，k_e 为一个常量。

[0010] $$l = \begin{cases} \dfrac{2}{1+e^{\frac{k_e}{\Delta_e}}} & if\ \Delta_e \neq 0 \\ 0 & if\ \Delta_e = 0 \end{cases} \quad \text{（公式 2）}$$

[0011] Kim and Lee 算法中选择具有较低清洁索引值的块首先擦除，既考虑了回收的效率，也考虑磨损的平衡，通过它来选择擦除块比贪婪算法更为准确高效。

[0012] 但是以上算法存在一个缺点，即当块的当前擦除次数接近 ε_{max} 时，该算法将忽略擦除次数，而偏重块的利用率，这是由于：

[0013] $$\lim_{\varepsilon_i \to \varepsilon_{max}} \frac{\varepsilon_i}{\varepsilon_{max}+1} = 1$$

[0014] 即当块的擦除次数越靠近最大擦除次数时，该极限越趋近于 1。这也就意味着对该块来说，其擦除次数在此清洁索引的算法中变为无效，清洁索引值仅由块的利用率 u_i 来判断。

[0015] 举个例子，取正规下齐平度 l 为 0.9，以下为采用 Kim and Lee 算法计算的清洁索引值：

[0016] 块 A：$u_i = 0.7$，$\varepsilon_i = 96000$，$\varepsilon_{max} = 96350$，$\varepsilon_{min} = 92950$

[0017] CleaningIndex = (1-0.9)×0.7+0.9×[96000/(96350+1)] = 0.96672

[0018] 块 B：$u_i = 0.4$，$\varepsilon_i = 96300$，$\varepsilon_{max} = 96350$，$\varepsilon_{min} = 92950$

[0019] CleaningIndex = (1-0.9)×0.4+0.9×[96300/(96350+1)] = 0.93952

[0020] 根据该算法，这时该选择清洁索引较低的块 B 擦除，而实际上它的 擦除次数比 A 还高，这反而加剧了 B 的磨损。

[0021] 以上这种情况下，使得高擦除次数，低利用率的块有可能被反复擦除，严重增加了其磨损，而这种情形在实际应用中肯定是不能被允许的。

[0022] 发明内容

[0023] 有鉴于此，本发明的目的在于提供存储块的擦除方法和装置，避免当擦除次数接近最大擦除次数时，出现的高擦除次数、低利用率的块会被反复擦除而严重增加其磨损的问题。

[0024] 为实现上述目的，本发明提供了一种存储块的擦除方法，包括以下步骤：

[0025] 根据含有无效数据的存储块的磨损均匀程度，动态调整回收效率与磨损平衡间的

倾向，获取所述存储块的清洁索引值，根据所述清洁索引值确定要擦除的存储块进行数据擦除。

[0026] 本发明还提供了一种存储块的擦除装置，包括：

[0027] 动态调整模块，用于根据含有无效数据的存储块的磨损均匀程度，动态调整回收效率与磨损平衡间的倾向；

[0028] 清洁索引模块，用于获取所述存储块的清洁索引值；

[0029] 擦除模块，用于根据所述清洁索引值确定要擦除的存储块并进行数据擦除。

[0030] 本发明通过采用改进的清洁索引算法，很好地解决了 Kim and Lee 算法中由于清洁索引计算方法方面的一些瑕疵，而导致的低利用率高擦除次数的块依据清洁索引选择出来，所造成的磨损加剧问题。它定义了一种新的清洁索引计算方法，引入了新的参量，使得当区块擦除次数接近擦除极限时，杜绝了 Kim and Lee 算法在该情况下的负面影响。

[0031] 同时，还提出了一种动态的调整方法，根据块的磨损均匀程度，动态调整回收效率与磨损平衡之间的倾向，保证闪存的垃圾回收中有较平衡的性能以及自适应能力。

附图说明

[0032] 图 1 为本发明的实施例中进行存储块擦除的方法流程图；

[0033] 图 2 为本发明实施例中一种存储块擦除装置的结构图。

具体实施方式

[0034] 由于 Kim and Lee 算法固有的一些瑕疵，导致低利用率高擦除次数的块依据清洁索引被选择出来，从而造成了磨损更为加剧。

[0035] 另外，Kim and Lee 算法中的正规下齐平度 1 是一个静态值，这使得其清洁索引值的计算被单一化，不能考虑到一些特殊的情况。如在芯片擦除次数较少时，应该尽可能地提高回收效率，即回收尽可能多的空间；而在芯片擦除次数较多时，就更应该考虑磨损平衡的因素。

[0036] 本发明的实施例通过采用改进的清洁索引算法，很好地解决了Kimand Lee算法中由于清洁索引计算方法方面的一些瑕疵，而导致的低利用率高擦除次数的块依据清洁索引选择出来，所造成的磨损加剧问题。它定义了一种新的清洁索引计算方法，引入了新的参量，使得当区块擦除次数接近擦除极限时，杜绝了 Kim and Lee 算法在该情况下的负面影响。

[0037] 同时，还提出了一种动态的调整方法，根据块的磨损均匀程度，动态调整回收效率与磨损平衡之间的倾向，保证闪存的垃圾回收中有较平衡的性能以及自适应能力。

[0038] 为使本发明的目的、技术方案和优点更加清楚，下面结合附图对本发明作进一步的详细描述。

[0039] 在本发明实施例中，根据改进的清洁索引方法进行存储块擦除的流程如图 1 所示：

[0040] 步骤 101、检查可回收块链表 dirtylist 是否为空，如果为空，则结束 回收；否则执行下一步。

[0041] 本发明实施例建立了新的存储含有脏数据－即无效数据－应被回收的块信息的数

据结构，在该数据结构里面保存有该块的清洁索引的值，并采用链表的方式存储，从而方便了增加或删除块信息，该链表称为dirtylist。并且不同的脏数据块信息按照清洁索引的值进行排序，值小的排在链表的前面。通过对此链表排序，在进行脏数据块擦除时，从此链表取出队首元素来擦除就可以了。该数据结构具体为：

[0042] struct dirtyblock

[0043] {

[0044] ……

[0045] uint cleanindex；

[0046] ……

[0047] }

[0048] 如果脏数据块链表为空，表明此时系统中没有可擦除的块，应结束本流程。

[0049] 同样，本发明实施例也定义了不含脏数据的非空闲数据块信息的数据结构，以如下结构体保存，这些信息也以链表形式存储，称为blocklist。其中free_size表示块的空闲空间，用来计算 u_i，erase_cnt用来记录擦除次数，当该非空闲数据块存储了脏数据需要擦除时，这两个值会被用来计算清洁索引值，并根据清洁索引值，放入dirtylist。

[0050] struct block

[0051] {

[0052] ……

[0053] uint32_t free_size；

[0054] uint32_t erase_cnt；

[0055] ……

[0056] }

[0057] 步骤102、计算系统的磨损倾斜度，即块的最大擦除次数和最小擦除次数之差，判断该差值是否超过阈值TH，如果超过则执行步骤103；否则执行步骤104。

[0058] 阈值TH称为磨损平衡阈值，磨损倾斜度反映整个系统所有存储块的磨损平衡程度，当这个值较大时，表明不同存储块的磨损程度相差较大，此时应当采用较大的1值，使得对清洁索引值的计算偏向于对擦除次数的考虑，也就是擦除次数值在清洁索引值的计算中占较大比重。反之，当磨损倾斜度值较小时，表明不同存储块的磨损程度较为均匀，此时应当采用较小的1值，使得对清洁索引值的计算偏向于对存储空间利用率－即块的使用率－的考虑，应尽量去回收使用率高的块。磨损平衡阈值TH就是为了衡量到底应优先考虑平衡块间的磨损，还是优先考虑回收存储空间。即，当磨损倾斜度大于TH时，选择较大1值；反之选择较小1值。

[0059] TH值是一个经验值，根据不同的应用场景而采用不同的设置，在本实施例中为2000。而较大和较小1值的确定，同样要根据具体应用，在本实施例中为0.9和0.1。

[0060] 步骤103、根据步骤102中的动态调整策略，计算当前情况下的正规下齐平度1，按计算出的正规下齐平度1，同时考虑块的擦除次数和块的利用率来计算每个dirtylist中的块的清洁索引值，并执行步骤105。因为超过了阈值TH，应选择较大1值。

[0061] 本发明实施例提出了一种改进的清洁索引的算法，通过对公式1进行修正，引入新的参量，来对清洁索引值作微调。改进后的清洁索引算法如公式3所示：

[0062] $$cleaningIndex = (1-l)u_i + l \times \frac{\varepsilon_i - \varepsilon_{min}}{\varepsilon_{max} - \varepsilon_{min} + 1}$$ （公式 3）

[0063] 其中，用 $\varepsilon_i - \varepsilon_{min}$ 代替 ε_i，使得对清洁索引的计算更为精确。修正后的清洁索引算法，避免了 Kim and Lee 算法的负面影响。

[0064] 依然以背景技术中的例子为例，块 A：$u_i = 0.7$，$\varepsilon_i = 96000$，$\varepsilon_{max} = 96350$，$\varepsilon_{min} = 92950$，因为 $\Delta\varepsilon$ 大于 2000，则选择较大 1 值 0.9，于是

[0065] CleaningIndex = (1-0.9)×0.7+0.9×[96000-92950/(96350-92950+1)] = 0.87711

[0066] 块 B：$u_i = 0.4$，$\varepsilon_i = 96300$，$\varepsilon_{max} = 96350$，$\varepsilon_{min} = 92950$

[0067] CleaningIndex = (1-0.9)×0.4+0.9×[96300-92950/(96350-92950+1)] = 0.92650

[0068] 根据该结果，块 A 会被先擦除。这就避免了 Kim and Lee 算法中擦除次数接近极限情况下出现的问题。

[0069] 可见，改进的清洁索引算法考虑了特殊情况下对清洁索引的计算，性能较为优秀，具有较大的实用价值。同时还提出了一种动态的调整方法，根据块的磨损均匀程度，动态调整回收效率与磨损平衡之间的倾向，保证闪存的垃圾回收中有较平衡的性能以及自适应能力。

[0070] 步骤 104、按计算出的正规下齐平度 1，同时考虑块的擦除次数和块的利用率来计算每个 dirtylist 中的块的清洁索引值。此时应应选择较小 1 值，来计算清洁索引值。

[0071] 步骤 105、对 dirtylist 中的每个块按计算出的清洁索引值进行从小到大的排序。

[0072] 步骤 106、从 dirtylist 的链表头节点中取出清洁索引最小的块信息，该块将被擦除并被回收，选择该块能够使得闪存的磨损差异尽量平均，尽管它的数据利用率比较高。

[0073] 步骤 107、将待擦除的块中的有效数据转移到一个空闲块上，并对原有数据作标记，标记为无效数据。

[0074] 在本发明实施例中，同样需要维护 freeblock 数据结构保存空闲块的块信息，保存在 freelist 链表中。用 badblock 保存擦除失败的块信息，保存在 badlist 中。Freeblock 和 badblock 的结构这里不再赘述。

[0075] 每次在闪存块内分配数据后，修改 blocklist 中的 free_size。当块内部有脏数据，满足可擦除条件时，利用 blocklist 中的 erase_cnt 和 free_size 可以计算出清洁索引的值，从而可以采用插入排序方法插入到 dirtylist 链表。

[0076] 步骤 108、由于原有应被擦除的块已经失效，这时通过对该块进行数据擦除，以回收作为空闲空间块使用。

[0077] 步骤 109、判断擦除是否成功，如果成功则执行下一步骤；擦除失败则该块被加入到 badlist 坏块链表，结束本流程。

[0078] 步骤 110、块的擦除已经成功，这时它的擦除次数加 1，用于该块被再次使用后，计算新的清洁索引值。

[0079] 步骤 111、块的擦除已经成功，该块已经转变为原始的空闲状态，写入 Cleanmarker 标记。Cleanmarker 标记是一种校验保护策略，用于标记块的擦除成功，可以直接写入新的数据。

[0080] 步骤 112、该块的各种标记已经标记完毕，这时已经可以被再次利用，被插入到 freelist，以供分配操作使用。

[0081] 本发明实施例通过采用改进的清洁索引算法，很好地解决了 KL 算法中由于清洁索引计算方法方面的一些瑕疵，而导致的低利用率高擦除次数的块依据清洁索引选择出来，所造成的磨损加剧问题。它定义了一种新的清洁索引计算方法，引入了新的参量，使得当区块擦除次数接近擦除极限时，杜绝了 KL 算法在该情况下的负面影响。

[0082] 同时，还提出了一种动态的调整方法，根据块的磨损均匀程度，动态调整回收效率与磨损平衡之间的倾向，保证闪存的垃圾回收中有较平衡的性能以及自适应能力。

[0083] 需要说明的是，在本发明的优选实施例中，可以对 1 值采用更为精确的动态调整策略，该动态调整策略为：

[0084] 通过进一步对 ε_i 分段，使得在擦除次数较低的情况下（低于一阈值），使用较大的 1，增大清洁索引值以避免被回收，来提高回收效率。高于所述一阈值时，同时根据 $\Delta\varepsilon$ 反映的磨损平衡程度，设置另一阈值来控制，在磨损较为均匀时，即低于另一阈值时，采用较小的 1，对清洁索引的计算更偏向空间利用率，而尽量去回收空间；在磨损均匀程度高于另一阈值时，采用较大的 1，使得对清洁索引的计算偏向于对擦除次数的考虑。

[0085] 图 2 为本发明实施例中一种存储块擦除装置的结构图，该装置具体包括：

[0086] 动态调整模块 21，用于根据含有无效数据的存储块的磨损均匀程度，动态调整回收效率与磨损平衡间的倾向；

[0087] 清洁索引模块 22，用于获取所述存储块的清洁索引值；

[0088] 擦除模块 23，用于根据所述清洁索引值确定要擦除的存储块并进行数据擦除。

[0089] 动态调整模块主要用于根据磨损倾斜度的值动态调整选择较大 1 值还是较小 1 值，清洁索引模块采用本发明实施例中的改进清洁索引算法计算各存储块的清洁索引值，确定擦除存储块的顺序。

[0090] 其中，所述动态调整模块 21 具体包括：

[0091] 判决单元 211，用于获得磨损倾斜度的值，判断该值是否超过一阈值；

[0092] 选择单元 212，用于根据所述判决单元的判断结果，如果该值超过所述阈值，则采用较大的正规下齐平度；否则采用较小的正规下齐平度。

[0093] 所述擦除模块 23 具体包括：

[0094] 排序单元 231，用于对含有无效数据存储块的链表中的每个存储块根据其清洁索引值进行从小到大的排序；

[0095] 定位单元 232，用于在所述链表的头节点中取出清洁索引最小的存 储块信息；

[0096] 数据迁移单元 233，用于将所述存储块中的有效数据转移到一个空闲块上，并对所述存储块的原有数据作标记，标记为无效数据；

[0097] 数据擦除单元 234，用于对所述存储块进行数据擦除。

[0098] 通过以上装置，能够实现当区块擦除次数接近擦除极限时，杜绝 KL 算法在该情况下的负面影响。

[0099] 同时，该装置还采用了根据块的磨损均匀程度，动态调整回收效率与磨损平衡之间的倾向，保证了闪存的垃圾回收中有较平衡的性能以及自适应能力。

[0100] 总之，以上所述仅为本发明的较佳实施例而已，并非用于限定本发明的保护范围。

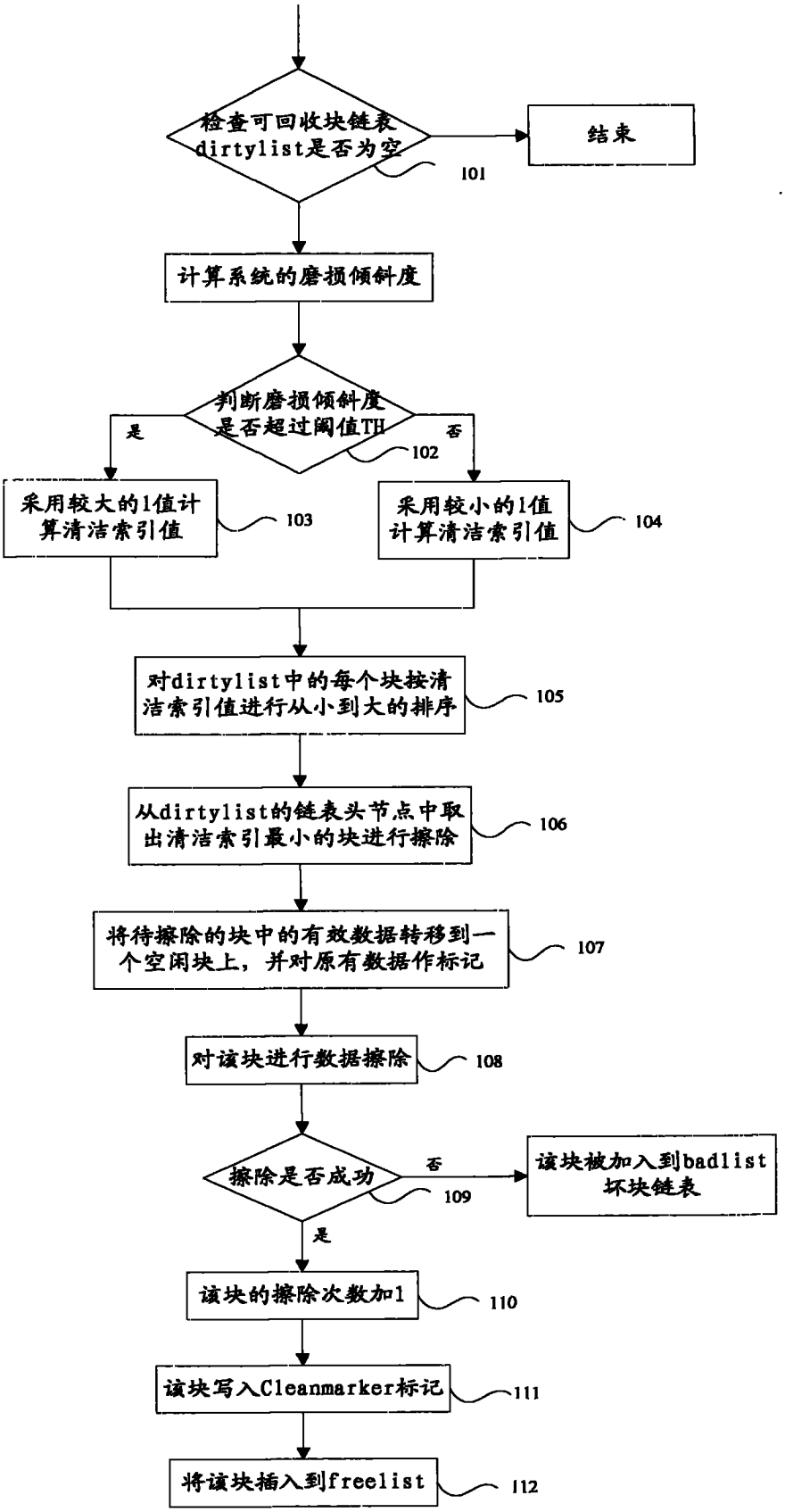

检查可回收块链表dirtylist是否为空
101
结束
计算系统的磨损倾斜度
判断磨损倾斜度是否超过阈值TH
102
是
否
采用较大的1值计算清洁索引值
103
采用较小的1值计算清洁索引值
104
对dirtylist中的每个块按清洁索引值进行从小到大的排序
105
从dirtylist的链表头节点中取出清洁索引最小的块进行擦除
106
将待擦除的块中的有效数据转移到一个空闲块上，并对原有数据作标记
107
对该块进行数据擦除
108
擦除是否成功
109
否
该块被加入到badlist坏块链表
是
该块的擦除次数加1
110
该块写入Cleanmarker标记
111
将该块插入到freelist
112

图 1

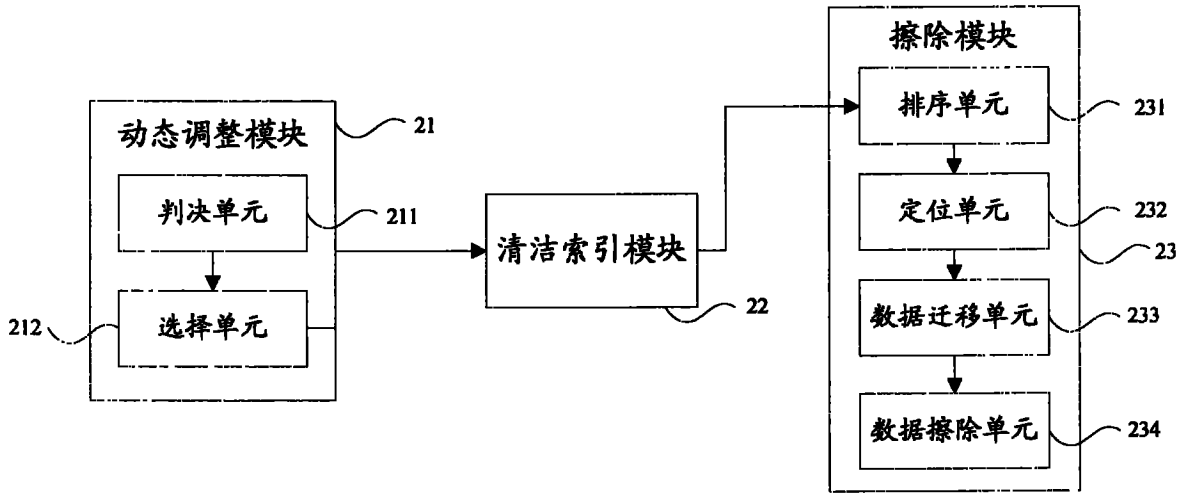

动态调整模块
21
判决单元
211
选择单元
212
清洁索引模块
22
擦除模块
排序单元
231
定位单元
232
23
数据迁移单元
233
数据擦除单元
234

图 2