JFFS3 闪存文件系统的分析与改进①

Research and Improvement of the JFFS3 File Systems

吕 莲 黄德才 (浙江工业大学 信息工程学院 浙江 杭州 310032)

摘 要： 对 Flash 的硬件特性和 JFFS3 文件系统进行了分析和研究。分析了 JFFS3 中垃圾回收策略存在的问题，在充分结合闪存自身读写特点并且不修改 JFFS3 基本 B+树结构的基础上，对 JFFS3 文件系统的垃圾回收策略提出了改进方案。

关键词： 闪存 JFFS3 文件系统 垃圾回收机制

1 闪存特性及闪存文件系统

Flash 是一种半导体存储器，具有系统掉电后仍可保留内部信息及在线擦写等功能特点。目前，flash 主要有：NOR(异或)和 NAND(与非)两种类型。NAND 写(编程)和擦除操作快，但随机存取慢； NOR 随机存取快，能以字节为单位写，但擦除慢。NAND 相对于 NOR，存储单元密度高、成本低、性价比好，适合大容量存储。NAND 由块(block) 构成，块由页(page) 构成，擦写以块为单位，读写以页为单位。每个块的擦除次数是有限的(NAND 型闪存上限约为一百万次左右)，当任何一个块达到擦除次数的上限时，都会影响整个闪存的性能。

传统的文件系统无法使闪存很好地工作，为了有效使用闪存，在过去的十年里人们进行了大量的研究[1]。JFFS2 是基于日志结构设计的闪存文件系统，日志节点是闪存设备上唯一的数据存储格式[2]。文件系统挂载时，JFFS2 扫描整个闪存并在内存中建立文件系统索引，挂载速度及内存消耗受到文件系统大小的影响。随着 NAND 闪存的出现以及闪存容量的快速增大，JFFS2 暴露出了其体系结构的很多问题，扩展性较差。

2 JFFS3日志文件系统

JFFS3 就是为解决 JFFS2 的这些问题而设计的，目前仍处于设计阶段。JFFS3 与 JFFS2 在设计上根本的区别在于，JFFS3 将索引信息存放在闪存上，而 JFFS2 将索引信息保存在内存中。JFFS3 文件系统的基本结构借鉴了 Reiser4 的设计思想，整个文件系统就是一个 B+树[3]。JFFS3 文件系统满足以下的要求：

· 内存消耗不必依赖于 FLASH 分区的大小；

· 提供文件系统快速加载，而不需要扫描整个闪存分区；

· 提供良好的垃圾回收和损耗均衡能力；

· 保证文件系统的断电可靠性等等；

2.1 JFFS3 文件系统结构

JFFS3 文件系统有两种类型的节点：索引节点(非叶节点)和叶节点。非叶节点具有固定大小，保存文件系统的索引信息。而叶节点具有可变大小，分为节点头和数据两部分。节点头中保存了对数据部分的总结

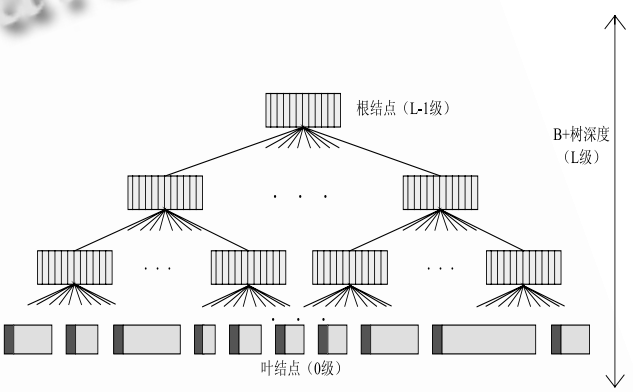


图 1 JFFS3 文件系统 B+树结构

① 收稿时间:2009-02-24

和描述信息。叶节点存储了文件系统的对象，这些对象包括：文件、目录项、扩展属性等等。整个文件系统的结构如图 1 所示。

2.2 日志处理

在 JFFS3 文件系统中，任何对文件系统的更改都会有一个新的节点被写入到闪存设备中，由此也引发了一系列对相关索引节点的更新，直至根节点。显然，如果对每次文件系统的更改都更新一系列索引节点，这样的代价太昂贵，JFFS3 提供了日志机制来解决这个问题。

日志由一系列的闪存擦写块组成，称为日志块。这些日志块在闪存中没有固定位置，任何闪存擦写块都可以被用作日志块。当文件系统发生改变时，对应的叶节点写入到日志块中，但是相应的索引节点不作更新，而是在内存中维护一个叫日志树的数据结构记录下这些更改，如图 2 所示。

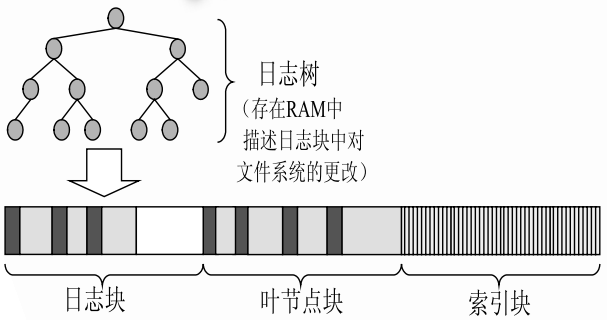


图 2　JFFS3 日志处理机制

2.3 垃圾回收机制和存在的问题

垃圾回收是所有日志结构文件系统中一个至关重要的部分。在 JFF2 中垃圾回收机制非常简单，步骤如下：1 从需要回收的块中移动有效节点到空闲块中；2 由于节点位置的变换，更新 RAM 中的索引信息；3 擦除回收块。JFFS3 的垃圾回收机制相对复杂。当有效数据被移动时，存储在闪存上的索引节点也必须相应地进行更新。这里就引发了一个问题：由于移动有效数据而造成闪存上索引的更新可能导致产生更多的垃圾页。这个问题可以由图 3 说明。

如图 3 所示，A 为子树的根节点，叶节点 D、E、F、G 位于编号为 1 的块中，索引节点 A、B、C 位于编号为 2 的块中。块 3、4 被保留用于垃圾回收，假定所有的节点和页大小一致。在状态 1 时，回收块 1 中的脏页，有效页被复制到空闲块 3 中，此时还不能将块 1 擦除，因为索引节点 B、C 仍然指向块 1，我们更新 B、C 索引节点，将他们复制到块 4 中，相应的根节点 A 也要被更新，如状态 3 所示。至此，块 1 可被擦除放入空闲块队列中，而块 2 中产生了 3 个新的垃圾页。由此可见，垃圾回收器的执行反而造成更多垃圾页的产生。

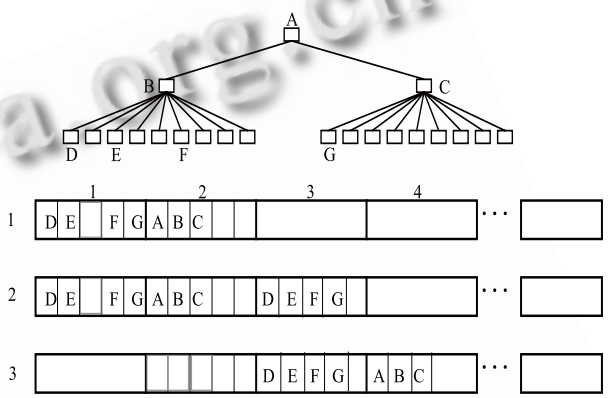


图 3　JFFS3 垃圾回收策略中存在的问题示例

3　垃圾回收机制的改进方案

3.1 闪存分区概念

为了解决 JFFS3 中垃圾回收可能导致更多垃圾产生的问题，我们对闪存进行分区，引入区域概念。一个区域由物理上连续的闪存块构成[4]，表 1 给了我们闪存大小，区域大小及他们之间关系的例子。

表 1　闪存大小、区域大小及两者之间关系的示例

闪存容量	块大小	块数量	区域中块数量	区域数量
8 G	128 KB	65536	8	8192
8 G	128 KB	65536	16	4096
8 G	256 KB	32768	8	4096

区域有三种类型：关闭、未关闭、空闲。关闭区域中所有的所有块都已使用，未关闭区域中有部分空间未使用，空闲区域中的所有块都未使用。为了提高文件系统访问效率，我们在每个关闭区域中添加本地索引信息，这有点像 JFFS2 中引入的 EBS 信息。未关闭区域中的本地索引信息保存在内存中直到区域被关

闭，我们保证未关闭区域数目尽可能小。

3.2 区域映射表

每个区域都有个标识它的逻辑区域号，我们在内存中维护一张逻辑区域到物理区域的映射表，如表 2 所示。

区域映射表由一组记录构成。第 n 条记录描述了第 n 个逻辑区域的信息，这些信息包括逻辑区域对应的物理区域编号，区域被擦写的次数，包含的脏页数以及区域的状态。映射表的更新存储在日志中，日志提交时这些更新也被提交了。

表 2　区域映射表

物理区域	擦写次数	脏页数	状态
0	21	12342	关闭
1	23	0	空闲
3	15	15544	关闭
2	29	3433	关闭

3.3 读操作实现

假定我们需要读灰色叶节点上的数据，如图 4 所示。

首先我们读取超级块，取得根节点的逻辑区域编号 0，然后查找区域映射表，获得逻辑区域编号为 0 的区域的物理区域号，这里为 1，我们看到这是个已关闭的区域，查找区域末尾附加的本地索引信息定位节点的物理地址。下个节点通过这个节点中存储的信息取得，读取下个节点的过程同上，直至所要读取的灰色节点。

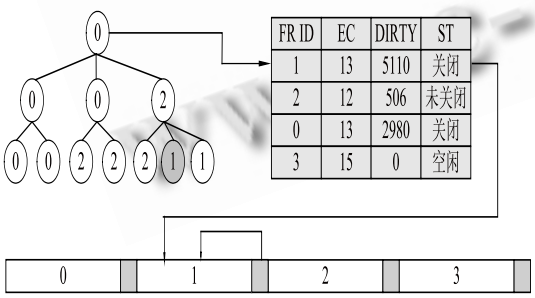


图 4　Flash 读操作

3.4 写操作实现

在选择区域写入新节点时，我们采用以下算法[5]：

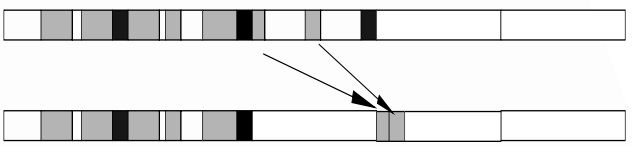
① 当存在未关闭区域时，选择空闲页最少的区域写入，以减少未关闭区域数量；

② 当没有未关闭区域时，选择空闲区域进行写入操作；

③ 当空闲区域数目少于 2 个时，后台执行垃圾回收。

3.5 垃圾回收机制实现

如图 5 所示，我们打算对物理区域 2 进行垃圾回收，物理区域 3 为空闲区域。在执行垃圾回收之前，物理区域 2 对应的逻辑区域编号为 2，物理区域 3 对应的逻辑区域编号为 3。首先，把物理区域 2 中的有效节点移动到区域 3 中；然后，修改 RAM 中的区域映射表，现在逻辑区域 2 对应了物理区域 3，并把物理区域 3 的擦除次数加 1，状态修改为未关闭；最后，对物理区域 2 进行擦除操作。



FR ID	EC	DIRTY	ST
1	9	10012	关闭
0	10	19540	关闭
2	9	28306	关闭
3	11	0	空闲

⇒

FR ID	EC	DIRTY	ST
1	9	10012	关闭
0	10	19540	关闭
3	11	0	未关闭
2	10	0	空闲

图 5　改进后的垃圾回收图示

4　性能测试

测试环境：1 硬件为 P4 1.7GHz (CPU) +256MB 内存；2 软件为 RedHat 9.0，Linux-2.6.26 版本内核[6]。基于以上系统环境使用 NandSim 模块模拟 NAND 闪存[7]，模拟参数设为：access_delay 10，program_delay100，erase_delay10，output_cycle 20，input_cycle 25。测试结果如表 3 显示，在闪存容量大小增加的情况下，内存消耗并没有急剧增加，在保证读速度大约为 20M/S 和写速度大约为 7M/S 的情况下，所需的内存消耗非常有限。

表 3　性能测试结果

闪存大小 MB	区域大小 KB	区域数量	读取 KB/S	写入 KB/S	消耗内存 KB
64	128	512	19705	7781	710
256	128	2048	20117	8015	1635
256	256	1024	20379	8203	981
256	512	512	18462	8102	805
512	256	2048	17990	7241	1714
512	512	1024	19890	7730	1014

5　总结与展望

本文对 JFFS3 文件系统进行了分析与研究。随着大容量闪存的发展，将文件索引转移到闪存上是必然趋势，笔者提出了对闪存分区域的概念，通过建立区域映射表来解决文件系统垃圾回收时，由于文件索引存在于闪存上而可能产生更多垃圾页的问题。目前 JFFS3 文件系统还处于研究阶段，其机制将需要进一步完善。

参考文献

1 Bar M,天宏工作室,译.Linux 文件系统,北京:清华大学出版社, 2003.21 - 142.

2 Woodhouse D. JFFS:The Journalling Flash File System, Red Hat, Inc.

3 Artem B. Bityutskiy dedekind@infradead.org, JFFS3 design issues, Version 0.32, November 27, 2005.

4 Gal E, Toledo S. Mapping structures for flash memories techniques and open Problems. IEEE International Conference on Software-Science, Technology and Engineering. 2005.83 - 92.

5 Kang JU, Jo H, Kim JS, Lee J. A superblock-based flash translation layer for NAND flash memory. Proc. of the 6th ACM&IEEE International Conference. 2006.161 - 170.

6 王学龙,等.嵌入式 Linux 系统设计与应用.第 2 版,北京:清华大学出版, 2001.56 - 107.

7 Claudia Salzberg Rodriguez,Gordon Fischer,Steven Smolski 著,陈莉君,贺炎,刘霞林,译.Linux 内核编程,北京:机械工业出版社, 2006.126 - 189.