第 36 卷 第 8 期 计 算 机 工 程 2010 年 4 月
Vol.36 No.8 Computer Engineering April 2010
·软件技术与数据库· 文章编号：1000—3428(2010)08—0052—03 文献标识码：A 中图分类号：TP311.5

基于 JBD 的日志型文件系统性能优化

陈 颖，奚宏生
(中国科学技术大学网络传播系统与控制联合实验室，合肥 230027)

摘 要：针对 EXT3 在嵌入式平台等易发生断电或系统崩溃的环境下频繁出现系统错误的问题，提出对日志块设备层(JBD)的改进方法，在不影响内核中其他功能前提下，采用同步写入的策略代替原始的异步缓冲机制，以提高文件系统的稳定性和应对上述突发事件的能力。实验结果表明，改进后的算法与原有算法相比，出错率明显降低。

关键词：Linux 系统；日志块设备层；文件系统；性能优化

Performance Optimization of Journaling File System Based on JBD

CHEN Ying, XI Hong-sheng
(Joint Lab of Network Communication System & Control, University of Science and Technology of China, Hefei 230027)

【Abstract】 Improvement of the Journaling Block Device(JBD), which replaces original mechanism of buffered asynchronous data writing with synchronous data writing, is presented in this paper for frequent system errors from EXT3 file system on the unstable platform like embedded system where there are often power failures and system collapses. It is meant to improve the stability and fault-tolerance ability for file system to handle emergency without affecting other kernel functions. Results of experiment show the obvious decrease of error rate.

【Key words】 Linux; Journaling Block Device(JBD); file system; performance optimization

1 概述

Linux 系统支持众多文件系统，包括从日志型文件系统到集群文件系统和加密文件系统。其中，作为 Linux 固有的文件系统，EXT2 文件系统是最为广泛采用的非日志型文件系统。但在日常使用中，断电故障或系统崩溃这样不可预测的事件可能导致文件系统处于不一致状态，为克服这一问题，采用自动进行的一致性检查 e2fsck，它将对整个磁盘进行全面彻底地检查和修复，并能有效地修复因断电或崩溃等异常带来的不一致问题，保证了文件系统的可用性。

磁盘容量的增大使这种检查和修复变得耗时费事。因此，EXT2 引入了日志块设备层(JBD)。JBD 通过将对文件系统的任何高级修改都分成两步进行。一般情况下，先把待写数据块的一个副本保存在日志中，当发往日志的 I/O 数据传送完成，即数据就提交到了日志时，数据块就开始写入文件系统。当发往文件系统的 I/O 数据传送终止时，即数据提交给文件系统了，日志中的块副本就被丢弃。当出现断电或是系统崩溃时，只需要让 e2fsck 对日志内容进行检查，然后对磁盘内容进行修复。而以日志为基础的修复磁盘的代价，相对于对整个磁盘进行检查修复，是微乎其微的。这样在向后完全兼容 EXT2 的基础上，可减少文件系统遇到断电故障或系统崩溃等不可预测事件时用于恢复文件系统一致性的所需时间。

经研究测试发现，在嵌入式平台这样频繁断电或系统崩溃一些恶劣的工作条件下，EXT3 文件系统也表现出了一些问题。如对以 EXT3 格式存储在磁盘上的文件进行正常的文件操作过程中若是出现了断电或是系统崩溃之类的情况，经 e2fsck 根据日志对文件进行检查修复之后，文件还是存在着丢失、不完整等问题。随着这些错误的累加，文件系统中的文件将越来越容易出现这些问题，出错率明显上升，导致最后唯一解决的办法就是将磁盘进行格式化，造成所有数据文件的丢失。这些问题严重地损害了 EXT3 文件系统对文件的保护性，危及到了 EXT3 文件系统的稳定性。这主要有 2 个原因：(1)与文件系统同在一个磁盘上的日志同样受到磁盘故障的困扰，也容易被损坏。(2)为提高效率而采用的异步缓冲写入机制使日志的事务在内存中停留的时间过长，更容易受到断电等故障的损害。

目前对日志型文件系统的研究可以分为 2 个方向：设计新的日志型文件系统，或是针对已有日志型文件系统进行性能优化。新日志型文件系统针对性较强，如闪存日志型文件系统 JFFS, JFFS2 等[1]，但移植性较差，使应用范围受到较大限制。已有的日志型文件系统如 XFS 和 JFS 等，都与已经得到大规模使用的 EXT2 文件系统不兼容。因此，深入研究 EXT3 文件系统的性能优化，提高 EXT3 文件系统的稳定性，具有十分重要的实践意义和理论意义。

针对日志型文件系统 EXT3，本文考察了 Linux 内核中虚拟文件系统(VFS)、EXT3 文件系统及 JBD 这几部分的相关代码和文献，研究了 JBD 的工作原理及算法，在此基础上提出了针对性的改进，以轻微的效率下降换取可靠性的大幅提高，并将其应用到嵌入式平台 μcLinux 上，进行了大量的测试，测试数据证实了这种改进的明显效果。

2 JBD 介绍及工作机制分析

2.1 JBD 基本数据结构

EXT2 和 EXT3[2]两者的区别主要在于 EXT3 加入了对

作者简介：陈 颖(1986－)，男，硕士研究生，主研方向：网络传播与控制；奚宏生，教授、博士生导师

收稿日期：2009-11-14 **E-mail：**baxt@mail.ustc.edu.cn

JBD 层接口的调用以及引入了日志的概念。JBD 有自己的数据结构：日志(journal)。依据文献[3]，其他文件系统也可以利用 JBD，但本文仅考虑 EXT3 文件系统。下文提到 JBD 时均特指 EXT3 文件系统的 JBD。

当 EXT3 文件系统以不同的方式挂载时，日志中存储的数据也不同，可能仅是元数据，也可能是数据和元数据都有，根据磁盘挂载模式而定。EXT3 文件系统不处理日志，日志对它是透明的。JBD 负责日志的创建等系列的操作。理论上，日志可以和文件系统处于不同的磁盘上，但一般情况下，日志和文件系统同处于一个磁盘中，此时，它与 EXT3 文件系统一样受到磁盘稳定性的限制，易受系统故障和磁盘断电的损害。

2.2 文件系统的不一致状态

EXT3 文件系统中的数据分为数据和元数(meta-data)据，见文献[2]。Linux 把所有数据都看成文件，目录也是种特殊的文件。元数据中记录了关于文件中数据的重要信息。文件系统的不一致状态，是指元数据中存储的信息与磁盘实际情况不一致。共有 6 种元数据：超级块(super block)，组描述符(group descriptor)，数据块位图(block bitmap)，索引节点(inode)及索引节点位图(inode bitmap)。这 6 种元数据均可从内核代码中可查找到，它们对文件系统至关重要，它们是文件系统对文件进行操作管理的基础和依据。

当出现断电或是系统崩溃导致文件系统存在不一致状态时，重启时 EXT2 会自动调用 e2fsck 对整个磁盘检查修复，这种修复是以整个磁盘上的元数据为基础进行的。

2.3 JBD 原始工作机制

早期这种处理方法是有效并可接受的。磁盘容量不大，这种重启后的检查修复所需的时间可以忽略。但随着磁盘容量的不断增长，所需时间越来越多，有时甚至长达一两个小时，损害了 EXT2 文件系统的可用性。

为缩短用于检查修复磁盘所需的时间，EXT3 借助于 JBD 来做到这点。JBD 会将对文件系统的修改，写入到日志中。这样在日志中记录了所有对文件系统的修改，若是出现了断电或是崩溃这样的故障时，重启时 EXT3 文件系统所需的只是对日志调用 e2fsck，根据日志的记录来检查和修复磁盘。

EXT3 文件系统和 JBD 之间相互独立，它们交互的基本单元有 3 个[2]：(1)日志记录：记录日志文件系统中一个磁盘块的一次更新；(2)原子操作处理(handle_s)：文件系统的一次系统调用所对应的日志记录，包含了多个日志记录，具有原子性，要么包含的日志记录全部完成，要么全不完成；(3)事务(journal_s)：包含多个原子操作处理，是 JBD 对磁盘写操作的单位。

一般情况下，JBD 如下工作，每次系统调用都会把若干对文件系统的修改以原子操作处理的方式打包，然后写入当前的事务中，当前的事务收集原子操作处理。每隔固定的时间间隔(5 s)，或是事务剩余空间不足，JBD 会将事务写到磁盘的日志中，如图 1 所示。

若出现断电或是系统崩溃，重启后自动进行的 e2fsck 检查可分为以下 2 种情况：若事务是完整的，则将日志中的事务内容重新写到磁盘上；若事务是不完整的，不论是该事务未完成，还是完成了，但没有顺利地完整写到日志中，事务中的内容均是无效的，e2fsck 无视它。这样就缩短了 EXT3 用于恢复系统一致性状态所需的时间。

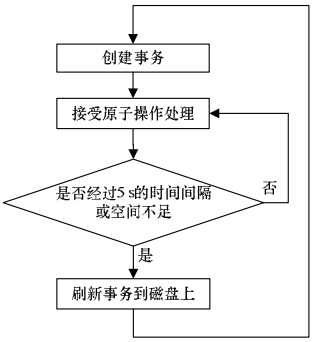

图 1 原始流程

EXT3 可以以 3 种不同的模式挂载，分别是预定、日志和写回，具体参见文献[2]。默认以预定模式挂载，保证效率和可靠性。

3 JBD 工作机制的缺陷及其改进措施

3.1 JBD 工作机制的缺陷

理论上，上述机制足以应付断电或是系统崩溃等状况，但在实际操作中存在着不足之处。在频繁地断电或是系统故障条件下，表现尤为明显。以下情况为预定模式挂载。

(1)在对同一小文件(2 MB 以内)进行操作过程中，事务空间未被写满，但是出现了断电或是系统崩溃等故障，这次日志中的事务是不完整的，此次的操作就会被丢弃掉。于是需要重复该操作，此过程中又出现断电或是系统崩溃故障。到一定次数后，就会出现文件系统错误，轻则该文件丢失，重则该文件出现 I/O 错误，不可读写，无法修改删除，也无法覆盖，必须重新格式化才能恢复正常；

(2)在对同一大文件(GB 级别)进行操作过程中，事务被完整地写到日志中。在出现断电或是系统崩溃等故障时，事务是完整的，此次操作未被丢弃，但文件的内容是不完整的，这是显然的。需要重复此操作，则会重新开始一项事务，而后若再次断电或是系统崩溃，此时也会出现和小文件类似的情况，轻则文件丢失，重则该文件出现 I/O 错误，不可读写，无法修改删除，也无法覆盖，必须重新格式化才能恢复正常。

出现这种问题的原因在于 EXT3 日志型文件系统存在以下缺陷：

(1)日志与文件系统存储在同一个磁盘上，这是最常见的情况。日志也是存储于磁盘上的文件，并不能免受磁盘故障的损害。在出现故障时，磁盘上的日志文件写入状态是不可保证的。若是作为恢复依据的日志内容出错，则在进行检查恢复时，也必定会出现错误，并且循环下去，出现错误的概率会越来越大，直到整个文件系统崩溃不可再用为止；

(2)事务是先缓存在内存中，等待一定时间间隔后(默认为 5 s)或是剩余空间不够了，才写到磁盘中的日志中。这样的话若是在写入磁盘之前出现断电，这些事务就会丢失。这样的话，会发生同上述情况一样的结果，作为恢复根据的日志内容不完整，或是有错。

EXT3 及 JBD 原有机制只是为了缩短用于磁盘检查修复所需的时间，但作为该机制核心基础的日志内容在一些频繁出现故障的情况下，仍然会出现错误，该机制就无法保证其可靠性，这在些极端情况下无法保证文件系统应有的可用性。

可以考虑将日志与文件系统两者所在磁盘分开，EXT3 文件系统支持这种做法。但这种做法不实际，因为存储于不同的磁盘的意义在于使用不同的电源。数个硬盘共用同一个

电源是非常常见的，出现了断电故障时几个硬盘全部会受到断电的损害。本文着重研究另一种做法：将日志同文件系统存储于同一磁盘上，但需要改进 EXT3 文件系统现有机制，以改善其性能。

经过仔细分析出错原因，可考虑将改变原有的异步缓冲写入机制，改为同步写入的方法，来改善现有机制。实验结果证明，这种做法的确取得了明显的效果。

3.2 JBD 工作机制的改进原理及措施

考虑到 JBD 层的数据结构在内存中保留的时间不固定，有时长达 5 s(默认)，这样就没办法保证在出现故障之前将数据刷新到磁盘上，写入日志中，就容易造成错误。因此，可在 JBD 层的系统调用级将原有的异步缓冲写入机制改为同步机制，在尽可能小地改变内核基础上，通过对内核进行少量的代码修改达到此目的。修改的核心思想为以效率的轻微损失换取可靠性的大幅度提升。修改后的流程如图 2 所示。

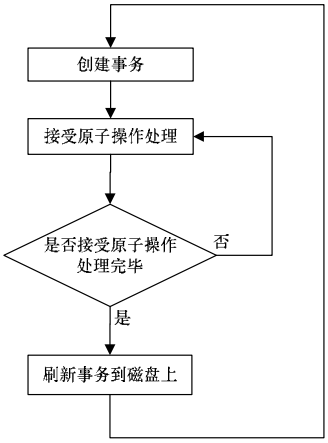

图 2　改进后的流程

修改可以分为以下几个要点：

(1)将元数据结构中的一些异步标志改为同步标志，这些标志表示元数据的写操作的属性。这样可以保证对元数据的创建修改等操作能够及时的生效。

(2)在 EXT3 中将日志中的事务主动地同步写入磁盘，而不是被动地等待固定的时候间隔或是空间不够了，这样 JBD 层的事务在内存中每次完成之后就会被立即写入磁盘，这样就减少了在内存停留的时间，降低了出错的概率。修改时必须调用 EXT3 封装好了的接口函数，而不宜越级调用内核级的函数。

具体修改如下：

(1) 修改 ext3/namei.c 中的 ext3_unlink, ext3_rmdir, ext3_mkdir, ext3_create 函数，在 ext3_journal_start 后加上 ext3_journal_force_commit(journal)。

(2)修改 fs/inode.c 中的 alloc_inode 函数，在 inode->i_flags = 0; 后添加 inode->i_flags|= MS_SYNCHRONOUS。

(3)修改 jbd/transaction.c 中的 new_handle 函数，在后添加 handle->h_sync=1。

4　测试数据

将修改后的 Linux2.4.22 内核重新编译并应用到高清媒体播放终端产品中(该终端使用了 Sigma Design 提供的高度集成的 EM8623L 芯片，其核心主控为主频 200 MHz 的 ARM 处理器，内存为 128 MB，希捷 IDE 接口的 7 200 转/分的 80 GB 硬盘)，在进行各种文件操作时断电测试。文件操作分别包括拷贝、删除 2 种操作，操作的对象有小文件(1 MB)、大文件(400 MB)2 种，断电的时刻分为操作未完成出现和操作刚完成后出现。限于设备和时间，只进行了拷贝操作的实验测试。

原始内核的测试数据见表 1。

表 1　原始内核测试数据

文件操作	总次数	完整的次数	文件消失的次数	不完整的次数	出现 I/O 错误的次数
拷贝过程中断电(400 MB)	100	0	42	19	39
拷贝过程中断电(1 MB)	100	0	56	1	43
拷贝完毕立刻断电(400 MB)	100	69	1	30	0
拷贝完毕立刻断电(1 MB)	100	15	55	15	15

由表 1 可见，原始的 2.4.22 内核版本中这些最常见的文件操作是非常容易被断电或系统故障所影响的。大文件的出错率在 2 种故障情况下达 81%和 31%，小文件的出错率达 99%和 85%。在嵌入式平台等一些不稳定的环境下，极其容易影响文件系统的可用性和可靠性。

针对添加修改之后内核进行同样的测试，测试数据见表 2。

表 2　改进后的内核测试数据

文件操作	总次数	完整的次数	文件消失的次数	不完整的次数	出现 I/O 错误的次数
拷贝过程中断电(400 MB)	100	0	0	100	0
拷贝过程中断电(1 MB)	100	1	0	98	1
拷贝完毕立刻断电(400 MB)	100	100	0	0	0
拷贝完毕立刻断电(1 MB)	100	100	0	0	0

由表 2 可见，改进后的内核版本中这些出错率大大降低了，基本上降为 0，这样就能说明这些改进的有效性。

5　结束语

本文从 EXT3 出现的文件系统错误出发，仔细考察内核中 EXT3 和 JBD 的相互作用机制，并提出针对性的改进，最后进行实验测试以确认这种改进。接下来的工作主要是验证这种改进后的内核的可靠性及其对效率的影响，包括：(1)文件删除操作的错误率。(2)文件夹拷贝和删除操作的错误率。(3)这种改进前后的磁盘速度的对比。以上是下一步需要进行的工作。目前情况下的测试都是在 EXT3 的挂载方式为预定模式下进行的，在日志和写回 2 种挂载模式下的情况也需要进一步的验证。希望本文能对其他日志型文件系统的改进有所启发。

参考文献

[1] 张　勇, 裘雪红. 嵌入式 Linux 下 JFFS2 文件系统的实现[J]. 计算机技术与发展, 2006, 16(4): 138-140.

[2] Bovet D P, Cesati M. Understanding the Linux Kernel[M]. 3rd ed. 陈莉君, 张琼声, 张宏伟, 译. 南京: 东南大学出版社, 2007.

[3] Sovani K. Linux: The Journaling Block Device[Z]. (2006-06-21). http://kerneltrap.org/node/6741.

编辑　金胡考