具有事件恢复功能的文件系统的研究与实现

李 涛[1,2] 梁洪亮[2]
(中国科学院软件研究所 北京 100190)[1] (中国科学院研究生院 北京 100039)[2]

摘 要 把一个进程在其执行过程中对文件的一系列修改称为一个事件。为了解决如何方便有效地恢复某个事件对文件的修改,研究并实现了一个具有事件恢复功能的文件系统GobackFS。GobackFS是一个用户空间文件系统,基于FUSE框架实现。测试结果表明,GobackFS文件系统在读写性能方面略低于Ext3文件系统,寻道速度方面略高于后者。

关键词 事件,文件系统,FUSE,Ext3

Design and Implement of Event-recovery File System

LI Tao[1,2] LIANG Hong-liang[2]
(Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[1]
(Graduate University, Chinese Academy of Sciences, Beijing 100039, China)[2]

Abstract This paper defined event as the modifications a process makes to a series of files during its running. In order to address the issue of how conveniently and effectively recover the modifications a event makes to files, the authors designed and implemented an event-recovery file system named GobackFS. GobackFS is a user space file system, and implemented based on FUSE. Test results show that GobackFS file system is slightly lower than Ext3 file system in read and write performance, and slightly higher than the latter in seeking speed.

Keywords Event, Filesystem, FUSE, Ext3

1 引言

在现代文件系统中,用户只能访问文件或目录的当前版本,而无法查看文件或目录的改变过程。文件系统在执行删除操作时会永久性地从存储介质中删除文件,在执行写操作时会永久性地改变文件的内容。用户可能无意地或错误地对文件执行了删除或修改操作,或者执行了一段恶意脚本,导致一系列文件遭到破坏。如果能够记录文件和目录的修改过程,可以把某次操作破坏的一系列文件全部恢复,这对用户是相当有帮助的。我们把一个进程在其执行过程中对文件的一系列修改称为一个事件。

文件系统可以通过文件版本化技术来记录文件的修改过程。文件版本化技术包括两种:快照技术和单文件版本化技术。快照技术周期性地建立文件系统的整个镜像或增量镜像,用户可以随时恢复到旧的镜像版本。这种技术最大的缺点是两个快照之间的改动无法恢复。单文件版本化技术是面向文件的恢复备份技术,在每一次写文件时创建新的文件版本。这两种技术都没有给用户提供一种方便有效的方法来撤销一个进程对多个文件的一系列修改,无法撤消一个事件对文件的破坏。

我们在单文件版本化技术的基础上提出了基于事件的文件系统恢复技术,并设计实现了一个具有事件恢复功能的文件系统GobackFS。本文组织如下:第2部分介绍了文件系统恢复技术的相关研究工作,第3部分讲述了GobackFS的设计目标,第4部分详细介绍了GobackFS的设计与实现,第5部分是GobackFS,CopyFS[13]和Ext3文件系统的性能对比测试与分析,最后是总结与展望。

2 相关工作

文件系统通过文件系统版本化技术来增强文件数据的可靠性和有效性。快照和单文件版本化这两种文件系统版本化技术都得到了广泛的应用。使用快照技术的文件系统有AFS[1],Petal[2]等。关于快照备份技术的调研和评估参见文献[3,4]。基于事件的系统恢复技术是在单文件版本化技术的基础上提出的,我们重点研究了实现单文件版本化的几种有代表性的文件系统。

Cedar[5]文件系统是第一个为单个文件保留版本的文件系统。文件版本在用户间共享,文件的每一次写操作创建一个新的文件版本。Elephant[6]文件系统是第一个为用户提供多种文件版本保留政策的文件系统。它基于FreeBSD 2.2.8内核实现。在第一次向打开的文件写数据的时候,Elephant透明地创建此文件的新版本。Elephant向用户提供了4种版

到稿日期:2008-04-08 本文受863重点项目,开源资源库协同开发技术及系统研究(2007AA010601),国家自然科学基金项目(60673022),国家科技攻关计划项目(2005BA113A02)资助。

李 涛(1983—),男,硕士研究生,主要研究方向为信息安全与系统软件,E-mail:litao19@gmail.com;梁洪亮(1972—),男,副研究员,硕士生导师,主要研究方向为系统软件和信息安全。

本保留政策：保留一份，保留全部，保留安全，保留标记版本。保留一份表示没有文件版本化，保留全部表示保留文件的所有版本，保留安全表示把文件版本保留一段时间，保留标记版本表示只保留重要的文件版本。用户可以标记某版本为重要版本。在 Elephant 文件系统中，用户还可以注册自己空间的资源回收政策。然而，Elephant 有其自己的类 FFS 的低级磁盘格式，是嵌入在 FreeBSD 内核中的 VFS，不能用在其他系统中。

VersionFS[7] 是基于 FiST[8] 的一种堆栈式文件系统，可以在多种文件系统上使用，并提供了多种用户配置政策。用户可以设置一个版本集的版本最大和最小数量，版本保留的最长和最短时间，以及版本可以占用的最大和最小空间。WaybackFS[9] 文件系统和 VersionFS 很类似，都是运行在一个现有的文件系统之上。WaybackFS 是基于 FUSE[10] 框架实现的，是第一个开源的版本化文件系统，不过没有向用户提供配置政策。

Ext3cow[11] 文件系统是基于 Ext3 实现的一个完整的文件系统，提供了快照和单文件版本备份双重功能。在 Ext3cow 系统中，用户可以直接看到某文件版本的内容。用户或应用指定一个文件名和某个时间点，Ext3cow 系统能够提供正确的快照和文件版本。Ext3cow 把快照和版本化功能完全封装在文件系统中，没有改变内核接口，也没有改变 VFS 提供的通用文件对象模型。所以，Ext3cow 很容易安装在现有系统中，可以作为运行时内核的模块加载，和其他的 Linux 文件系统共存。

总之，基于现有技术的文件恢复系统提供了强大的系统恢复能力和多样化的备份策略和恢复策略，但是没有提供一种有效的方法恢复单个进程对多个文件的一系列修改。比如，用户无意执行了一段病毒脚本，破坏了一系列文件。在使用单个文件版本化的系统中，用户很难确定病毒破坏了哪些文件，应该恢复哪些文件。如果采用快照技术，可以恢复到上一个可用的快照，那么此快照创建后被正常修改或者新增加的文件也会被撤销。所以，本文提出了基于事件（如病毒破坏）的系统恢复机制，用户可以方便地恢复某事件破坏的文件，而不会影响正常的文件。

3 GobackFS 文件系统的设计目标

我们研究和设计了一个具有事件恢复功能的文件系统 GobackFS。在 GobackFS 文件系统中，文件在每一次打开和关闭期间如果发生了改变，就创建一个新的文件版本，并记录促使新版本创建的进程信息。GobackFS 文件系统的设计目标有 3 个：简单易用、方便移植、易于实现。

3.1 简单易用的用户接口

GobackFS 文件系统的使用界面和通用文件系统（如 Ext3）一样，GobackFS 对底层的备份数据进行了封装。在 GobackFS 文件系统中，用户可以通过一条简单的命令来查看最近某段时间内哪些进程对文件做了更改，某个进程对哪些文件做了更改，并且通过一条命令就可以撤销某个进程对一系列文件的更改。用户还可以方便地查看单个文件的所有保留版本，并可以恢复到任意版本。

3.2 用户空间文件系统

实现一个具有特殊功能的文件系统，开发者常常从以下几个层次入手：设备驱动层、系统调用层（虚拟文件系统层）及应用层等。为了减少开发的工作量，开发者可以选择一个现有的文件系统作为起点，通过修改它的源代码和底层的设备驱动程序，把需要的功能加进去。但是这种开发方式存在很多问题：首先，开发者必须理解现有文件系统的源代码组织结构；其次，修改过程中可能会调用当前操作系统内核中特有的代码，从而牺牲了文件系统的可移植性；再次，内核文件系统的调试工作比较复杂，降低了开发的效率。

开发者可以选择在用户层开发文件系统。这种文件系统工作在用户空间，便于开发和移植，当然，由于存在大量内核空间和用户空间数据的拷贝和上下文的切换，它的工作性能会比较低。

由于用户空间文件系统的实现相对简单，我们可以专注于基于事件的文件恢复功能的实现，并且这种文件系统方便移植到许多现有文件系统上。所以，我们在用户空间实现 GobackFS 文件系统。

3.3 基于 FUSE 框架实现

用户空间文件系统的框架主要有两个，LUFS（Linux Userland File System）[12] 和 FUSE（File system in UserSpacE）[10]。目前，LUFS 社区不够活跃，用户也越来越少，最新版本还是在 2003 年发布的。相反，基于 FUSE 框架的应用却越来越多，目前已经相当稳定，并且进入了 Linux 内核官方 2.6 版本。

FUSE 在用户空间提供了文件系统的能力，它的通讯接口设计得非常简单、有效、安全，能够支持大多数的文件系统语义。使用 FUSE 框架，避免了内核级的开发，降低了开发难度，我们可以更加专注于版本化功能的开发。

4 GobackFS 文件系统的设计与实现

GobackFS 文件系统的设计参考了开源文件系统 CopyFS[13] 的架构，CopyFS 实现了简单的单文件版本化功能，记录了文件的修改过程。为了记录进程的信息，支持基于事件的系统恢复功能，我们对 CopyFS 的架构进行了较大的修改。

4.1 记录文件版本信息和进程信息的数据文件

在 GobackFS 文件系统中，用户看到的每个文件（目录也看作文件）都对应一个 metadata 文件，记录这个文件的版本信息和触发此文件新版本创建的事件 id，文件名为"file_name"的文件对应的 metadata 的文件是"metadata. file_name"。metadata 中的每一行表示此文件某版本的信息，最后一行是当前版本的信息。metadata 文件的每一行的格式为"eventid：vid：svid：mode：uid：gid：time：name"，eventid 表示触发此版本创建的事件 id，用 8 位的 16 进制数表示，vid 表示文件版本号，svid 表示文件子版本号，mode 表示文件的访问权限，uid 表示文件拥有者 id，gid 表示文件拥有者所在组 id，time 表示此版本的创建时间，是从公元 1970 年 1 月 1 日 00：00：00 以来经过的秒数，name 表示此文件版本的文件名。

我们把触发文件版本创建的事件，即修改文件的事件的信息，保存在一个事件信息文件"events_info."中。每行表示一个事件，事件信息包括 eventid、对应的进程名、对应进程的 pid、对应进程的启动时间和系统启动时间，每一项也是用"："隔开。这里的进程启动时间是从系统启动开始，时间流过的滴答数。

GobackFS 文件系统对文件版本信息文件和事件信息文件进行了封装，用户察觉不到这些文件，也无法直接访问这些文件。

4.2 文件版本的创建

在 GobackFS 文件系统中，文件的每个版本保存在单独的文件中。文件名"file_name"的文件的第 N 个版本保存在文件"N. file_name"中，其中，N 是 8 位的 16 进制版本号。文件"file_name"的最初版本是"00000000. file_name"。我们对这些数据信息进行了封装，用户只能看到文件的当前版本。

6 个文件操作创建新的文件版本，分别为：unlink，rmdir，mknod，mkdir，truncate，open with WRITE flag。两个文件操作创建新的文件子版本，分别为：chmod 和 chown。

unlink()系统调用删除一个文件，rmdir()系统调用删除一个目录。执行这两个系统调用时，在文件或目录对应的 metadata 文件的最后添加一行"eventid:0:0:0000:0:0:0:"，它的 vid 项是 0，表示此文件或目录已被删除，这种文件会被隐藏起来。GobackFS 并没有执行真正的删除操作，用户可以再次恢复这些文件和目录。

mknod()系统调用创建普通文件结点，mkdir()系统调用创建目录。这里有 3 种情况需要考虑，第一种是待创建文件或目录的名称与同目录下的文件名或目录名冲突，直接向用户返回错误就可以了。第二种是待创建的文件名或目录名与同目录下已被删除的某文件名和目录名冲突，被删除的文件并没有真的被删除，只是被隐藏起来了。在这种情况下，我们把新创建的文件或目录看成被删除文件或目录的一个新的版本，把原 metadata 文件的最后一行"eventid:0:0:0000:0:0:0."删除，并添加此新版本的信息。第三种情况下不存在名称冲突问题，我们直接创建文件和目录，初始化其版本信息就可以了。

当 truncate()系统调用改变文件大小和以 WRITE 标记打开一个文件时，我们也会为文件创建一个新版本，以保护文件原来的数据。我们还支持为文件或目录建立符号链接的操作。

chmod()系统调用和 chown()系统调用是改变文件或目录访问权限和所有权的操作，GobackFS 还记录了文件和目录访问权限和所有权的改变过程。这两个操作会创建新的文件子版本，子版本号(svid)在原子版本号的基础上加一，主版本号(vid)不变。

在创建新版本时，记录触发此次版本创建的事件对应的进程 pid，进程的启动时间和系统的启动时间。pid 可以通过函数 fuse_get_context()->pid 获得。进程启动时间可以从/proc/[number]/stat 中读出。系统启动时间可以从/proc/stat 文件中读取。把此事件与事件信息文件"events_info."中的事件对比，如果 pid 和某事件对应进程的 pid 相同，进程启动时间相同，并且系统启动时间相同，那么这两个事件是同一事件，可以得到此事件的 eventid 等信息，创建新版本时把 eventid 写入文件的 metadata 文件中。否则，事件信息文件中没有和此事件相同的事件，那么，我们为此事件分配一个 eventid，并把该事件的 eventid、对应的进程名、对应进程的 pid、对应进程的启动时间和系统启动时间写入事件信息文件中。

4.3 基于事件的文件恢复过程

当用户恢复 eventid 为 id 的事件对一系列文件的修改时，GobackFS 递归遍历所要查找的目录中的 metadata 文件，找到 eventid 为此 id 的文件版本，然后把此版本的上一版本置为当前版本。当用户或进程对文件进行操作时，此文件或目录下的 metadata 信息会读到内存中，以加快文件的存取速度。所以，通常情况下，在用户执行文件恢复功能时，很多要查找的目录或文件的 metadata 信息已经在内存中了，加快了文件的访问速度。另外，用户还可以查看某个文件和目录的所有版本，方便地恢复到任意版本。

4.4 GobackFS 文件系统的实现

GobackFS 文件系统基于 FUSE 2.7.2 版本实现。由于 FUSE 对于几乎所有的文件系统调用都有默认实现，因此我们只需要重写我们关心的文件系统调用就可以了。我们重新实现了 24 个文件系统调用，共 3200 行的 C 代码和近 800 行的脚本程序，实现了设计的所有功能。

GobackFS 把相关文件的版本信息和事件信息读入一个哈希表中，明显加快了版本信息的创建速度和文件的恢复速度。当此哈希表占用的内存达到一定的数量后，GobackFS 会自动释放一部分最早读入哈希表中的版本信息和事件信息，以确保文件系统不会占用过多的内存。

5 性能分析

我们对 GobackFS，CopyFS 和 Ext3 文件系统进行了性能对比测试，选择 CopyFS 参与对比测试是由于 GobackFS 参考了 CopyFS 的架构，CopyFS 也是基于 FUSE 框架编写的，实现了简单的文件版本化功能。

我们选择 Bonnie[14] 进行性能测试。Bonnie 是一个著名的测试 Unix 文件系统性能的测试工具。Bonnie 有两个主要参数，一个是待测试的文件大小，另一个是此文件的期望存储路径。然后 Bonnie 创建这个大文件，对其进行一系列操作，包括以字符为单位写、以块为单位写、重写(rewriting)、以字符为单位读，以块为单位读和寻道(seeking)。在本测试中，我们设置的测试文件大小是 1GB。

我们使用的测试环境如下：操作系统是 Ubuntu 7.10，硬件配置为：CPU P4 2.4GHz，内存 512M，硬盘 60G。Bonnie 不同阶段的性能测试结构如图 1 至图 4 所示。读写速度测试单位是 kB/s，寻道测试的单位是次数/s。

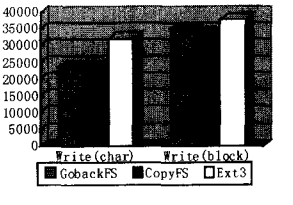

图 1 写操作性能对比测试结果

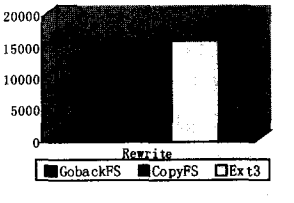

图 2 重写操作性能对比测试结果

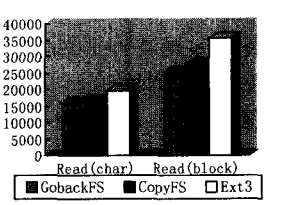

图 3 读操作性能对比测试

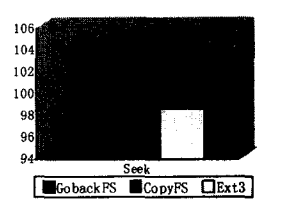

图 4 寻道操作性能对比测试结果

(下转第 280 页)

表 2　两次实验数据

	f(错误数)	n(访问量)
实验 1	10230	267801
实验 2	20153	1755152

采用 Nelson 模型，实验 1 和实验 2 对该 Web 软件的可靠性评估计算结果分别为：

Reliability1＝1－13212/878323＝96.18%

Reliability2＝1－28336/1755152＝95.47%

将两种统计方法得到的 Web 软件可靠性(R)对比，结果如表 3 所示。

表 3　实验数据对比

	实验 1	实验 2
马尔可夫链使用模型	96.18%	95.47%
UMMs	98.50%	98.39%

通过实验数据对比我们可以发现，采用本文提出的基于 UMMs 的统计测试方法能够对 Web 软件进行更详尽的测试，提高测试的覆盖率和软件的可靠性，能够满足复杂 Web 应用软件的统计测试。

结束语　本文将 Web 软件统计测试方法应用于 SWU/CS Web 软件，分析该软件的 Web 日志信息，构建该 Web 软件的 UMMs。为了评估该软件的可靠性，我们将统计得出的访问数据和错误数据用于 Nelson 模型，计算得出该软件的可靠性值和 MTBF，以此证明该软件是稳定的和可靠的。同时也说明 UMMs 对于 Web 软件的统计测试使用模型是合适的，多重的 UMMs 构建对于判断一个 Web 软件的可靠性更有必要。最后将该方法和基于马尔可夫链模型的统计测试方法进行对比，证明该方法能够提高测试的覆盖率和软件的可靠性。

参 考 文 献

[1] Kallepalli C, Tian J. Measuring and modeling usage and reliability for statistical Web testing. IEEE Trans. Software Eng., 2001, 27(11): 1023-1036

[2] 沈宏. 基于用例和日志的 Web 统计测试. 硕士学位论文. 上海师范大学, 2004

[3] Tian J, Ma L, Li Z, et al. A hierarchical strategy for testing Web-based applications and ensuring their reliability // Proc. of the 27th Annual International Computer Software and Applications Conference. 2003

[4] 张慧，熊前兴. 基于 UML 的软件统计测试方法研究[J]. 计算机与数字工程, 2008, 36(1): 22-22, 25

[5] 邓晔，徐锡山，颜炯. 基于 UML 的软件使用模型的研究及工具实[J]. 计算机应用研究, 2006, 23(1): 129-131

[6] Hao Jian-hua, Emilia M. Usage-based Statistical Testing of Web Applications. ICWE'06, Palo Alto, California, USA, July 2006

[7] 路晓丽. Web 应用软件的测试技术研究. 西北大学博士学位论文. 2006

[8] Walton G H, Poore J H. Statistical testing of software based on a usage model. Software-practice and Experience, 1995, 25: 97-108

[9] Karlin S, Taylor H M. A First Course in Stochastic Processes, second ed. New York: Academic Press, 1975

[10] Whittaker J A, Thomason M G. A Markov Chain Model for Statistical Software Testing. IEEE Trans. Software Eng., 1994, 20(10): 812-824

[11] Tian J, Lin E. Unified Markov for Models Software Testing, Performance Evaluation, and Reliability Analysis // Proc. Fourth ISSAT Int'l Conf. Reliability and Quality in Design. Aug. 1998

[12] Tian J, Nguyen A. Statistical Web Testing and Reliability Analysis // Proc. Ninth Int'l Conf. Software Quality. Oct. 1999: 263-274

(上接第 272 页)

从测试结果可以看出，GobackFS 文件系统和 CopyFS 文件系统的性能基本相同，在读写方面都略低于 Ext3 文件系统，但在寻道操作方面均优于 Ext3，这可能和 FUSE 框架的底层实现有关。

结束语　我们把一个进程在其执行过程中对文件的一系列修改称为一个事件。为了解决如何方便有效地恢复某个事件对文件的修改，我们研究并实现了一个具有事件恢复功能的文件系统 GobackFS。GobackFS 文件系统基于 FUSE 框架实现，是一个用户空间的文件系统，可以方便地 mount 到通用 Unix/Linux 文件系统上使用。测试结果表明，GobackFS 文件系统的读写性能略低于 Ext3 文件系统，但寻道速度略高于后者。

目前，GobackFS 文件系统提供的功能还比较简单，我们下一步的工作主要包括两个方面：第一，为用户提供差分备份的功能和磁盘空间的控制功能。第二，用户可以选择性地备份文件，比如用户可以选择不备份某些格式的文件，如 mp3 和 avi 格式等。

参 考 文 献

[1] Kistler J J, Satyanarayanan M. Disconnected operation in the Coda file system // Proceedings of 13th ACM Symposium on Operating Systems Principles. Asilomar Conference Center, Pacific Grove, CA, ACM Press, October 1991: 213-225

[2] Lee E K, Thekkath C A. Petal: Distributed virtual disks // Proc. of the 7th Conference on Architectural Support for Programming Languages and Operating Systems. 1996

[3] Chervenak A, Vellanki V, Kurmas Z. Protecting file systems: A survey of backup techniques // Proceedings of the Joint NASA and IEEE Mass Storage Conference. 1998

[4] Azagury A, Factor M E, Satran J. Point-in-time copy: Yesterday, today and tomorrow // Proceedings of the Tenth Goddard Conference on Mass Storage Systems and Technologies. 2002: 259-270

[5] Gifford D K, Needham R M, Schroeder M D. The Cedar File System. Communications of the ACM, 1988, 31(3): 288-298

[6] Santry D S, Feeley M J, Hutchinson N C, et al. Deciding When to Forget in the Elephant File System // Proceedings of the 17th ACM Symposium on Operating Systems Principles. December 1999: 110-123

[7] Muniswamy-Reddy, et al. A Versatile and User-Oriented Versioning File System // Proc. of the 3rd USENIX Conference on File Storage and Technologies. March 2004

[8] Zadok E, Nieh J. FiST: A Language for Stackable File Systems // Proceedings of the Annual USENIX Technical Conference. June 2000: 55-70

[9] Cornell B, Dinda P A, Bustamante F E. Wayback: A user-level versioning file system for Linux // Proceedings of the USENIX Annual Technical Conference. FREENIX Track. 2004: 19-28

[10] Szeredi M. Filesystem in USEr space. http://sourceforge.net/projects/avf, 2003

[11] Peterson Z, Burns R. Ext3cow: A Time-Shifting File System for Regulatory Compliance. ACM Transactions on Storage, 2005, 1(2): 190-212

[12] http://sourceforge.net/projects/lufs/

[13] Vigier N, Joubert T. http://n0x.org/copyfs/

[14] Bray T. The Bonnie Benchmark. http://www.textuality.com/bonnie