HV-Recovery：一种闪存数据库的高效恢复方法

卢泽萍 孟小峰 周 大
（中国人民大学信息学院 北京 100872）

摘 要 和磁盘相比，闪存作为一种新型的存储设备，具有读写速度快、抗震、省电、体积小等优点. 因此，当前的研究普遍认为闪存将取代磁盘成为新一代的数据库二级存储设备. 但是，由于闪存具有和磁盘不同的一些固有的读取特性，将当前基于磁盘设计的数据库直接移植到闪存上时，并不能充分发挥闪存设备的优越性. 在数据库的恢复过程中，由于闪存的异地更新和重写之前先擦除的特性将带来大量高代价的小的随机写，直接使用传统的恢复方法在闪存数据库中就更难以充分利用闪存的优越性. 因此，文中提出了一种对闪存中天然存在的数据的历史版本来进行管理和利用的恢复方法 HV-recovery，来改进 undo 恢复的性能. 通过和开源数据库 Oracle Berkeley DB 的比较，实验结果表明 HV-recovery 是原有的恢复算法性能的 2～8 倍，充分说明了其优越性.

关键词 闪存；闪存数据库；固态硬盘；恢复；日志
中图法分类号 TP311 **DOI 号**：10.3724/SP.J.1016.2010.02258

HV-Recovery: A High Efficient Recovery Technique for Flash-Based Database

LU Ze-Ping MENG Xiao-Feng ZHOU Da
(*School of Information, Renmin University of China, Beijing* 100872)

Abstract Flash memory, as a new kind of data storage media, has a lot of attractive characteristics when compared with Hard Drive Disk (HDD) such as fast access speed, shock resistance, power saving, lighter form and low noise. Therefore flash memory is considered as the main storage device instead of disk in the next generation. However, traditional disk-based database can't take full advantage of high I/O performance of flash memory if we transfer it to flash memory without modification. The main reason is flash memory embraces different access characteristics with HDD. As for recovery, the situation becomes more serious because the out-of-place update model and erase-before-rewrite of flash memory lead to high cost of large quantity of minor random writes during the course of recovery. This paper proposes a recovery method, HV-recovery, to improve the performance of undo. HV-recovery makes use of the history versions of data which is naturally emerged in flash memory due to the out-of-place update. Experimental results on Oracle Berkeley DB show that the HV-recovery outperforms traditional recovery in 2×～8×. The results demonstrate the high efficiency of the method.

Keywords flash memory; flash-based DBMS; SSD; recovery; logging

1 引 言

随着信息技术的飞速发展，数据呈爆炸性增长，海量的数据对数据库系统性能的要求也越来越高. 而作为当前比较主流的二级存储介质，磁盘因为其内部的机械移动已经成为 IO 性能的瓶颈，越来越不能满足实际应用系统对数据存取带宽的需求. 在

收稿日期：2010-06-11. 本课题得到国家自然科学基金（60833005，60573091）、国家“八六三”高技术研究发展计划项目基金（2007AA01Z155，2009AA011904）、教育部博士点基金项目（200800020002）资助.**卢泽萍**，女，1985 年生，硕士，主要研究兴趣为闪存数据库系统. E-mail：luzeping_july@yahoo.com.cn. **孟小峰**，男，1964 年生，教授，博士生导师，主要研究兴趣为 Web 数据管理、XML 数据库、移动数据管理. **周 大**，男，1980 年生，博士，主要研究方向为闪存数据库存储、索引和事务处理.

过去的 20 年里，CPU 处理速度增加了 570 倍，而磁盘的访问速度却只增加了 20 倍[1]. 可见，CPU 和主要二级存储器磁盘之间的带宽鸿沟已经成为了制约计算机系统处理能力提高的主要瓶颈.

值得庆幸的是，闪存作为一种新型的固态存储设备，由于其读写速度快、消耗电量低、抗震、小巧轻便等优点，已经受到越来越多的关注. 随着容量的不断增大和单位价格的不断下降，许多研究者纷纷预测闪存将逐渐取代磁盘成为新的主流二级存储设备. 图灵奖得主 Gray Jim 在 2005 年就曾预测说“就像磁盘取代磁带一样，闪存将会取代磁盘”[1]. 即使对现有的数据库不做任何改进，直接移植到闪存上，其性能也能提高大约 10 倍[2].

但是，由于闪存其固有的特性，若将现有的面向磁盘的传统数据库直接运行在闪存存储器上，还不能充分发挥闪存的优越性. 因此，当前迫切需要将传统的数据库进行改进，让其更好地适应闪存本身的特点，以进一步提高闪存数据库的性能[3-4].

数据库开发和应用的实践表明，数据库恢复技术作为数据库系统中不可缺少的组成部分，对整个系统的性能影响是非常大的[5-7]. 在恢复过程中，为了对事务已经更改的数据项进行还原，通常需要对数据库中的一些已经赋予新值的数据进行重写. 而这种大量的小随机重写操作，对闪存的代价是非常巨大的，不但浪费空间，而且非常耗时. 因此，迫切需要一种高效且稳定性强的闪存数据库的恢复技术.

本文针对闪存存储器中天然存在的历史版本数据，提出了一种充分利用这些数据的历史版本从而进行恢复的一种新型的恢复方法 HV-recovery. 总的来说，本文所做的主要贡献如下：

(1) 本文研究了闪存数据库中的恢复问题，并提出了新的适用于闪存的恢复方法.

(2) 提供简单有效的恢复操作. 有效地减少在恢复过程中容易出现的冗余写操作，从而大幅度减少恢复时间.

(3) 优化日志结构. 减少过多的日志冗余，从而提供高效的日志文件.

(4) 提高空间利用率. 减少大量垃圾数据的存在，从而提高存储设备中的空间利用率.

本文第 2 节介绍闪存特殊的物理特性给恢复带来的挑战以及相关工作；第 3 节详细介绍本文设计的 HV-recovery 的基本原理；第 4 节提出怎样针对 HV-recovery 中的设计进行进一步的性能优化；第 5 节用分析及实验结果证明设计的优越性；最后第 6 节进行总结.

2　问题定义及相关工作

闪存和磁盘读写特性的不同使得将现有的传统的数据库移植到闪存上时会出现一些问题. 下面将具体介绍在恢复中出现问题的原因和现有的一些改进方案及它们所存在的问题.

2.1　闪存存储器的物理特性

没有机械延迟. 我们知道，在磁盘中，访问数据的时间主要用于移动磁头以及等待磁盘旋转. 而闪存没有像磁头一样的机械部件，其随机访问模式和顺序访问模式的开销是相当的. 这样，就可以把数据离散地分布，这并不会使访问的开销增加.

重写之前先擦除. 众所周知，在磁盘中，如果需要更新数据，这些数据的新版本可以直接原地覆盖在旧版本所占有的地址上，这就是所谓的原地更新. 可是在闪存中，在数据的旧版本没有被擦除前，是不能在原地写入新的版本的. 也就是说，如果修改一个数据，就需要对整个块(通常为 64K 或 128K)上的数据进行擦除，这是非常巨大的代价. 因此，在闪存中往往会采取异地更新的方式，即把数据的新版本写入另外的空闲空间中，而不直接在原地覆盖.

读写速度不一致. 在闪存中，不同的访问操作的速度差别很大. 一般来说，读的速度很快，写的速度略慢，擦除的速度最慢. 因此，在设计新的基于闪存的数据结构中，应当尽量减少写操作和擦除操作，可以适当增加读操作，以整体上提高系统性能.

有限的擦除次数. 虽然闪存中的块是可以进行反复擦除的，但每个块的擦除次数是有限的，一般为 10000～100000 次. 因此，就必须尽量减少写入的次数，以间接减少擦除的次数，来延长闪存的使用寿命.

2.2　问题定义

在面向磁盘的数据库系统中，基于日志的恢复技术被广泛采用. 不同的协议之下，日志记录的设计、日志/数据缓冲区的管理、检查点机制、记录日志和恢复的过程都很不一样. 以最为常见的 undo 日志为例，当事务 T 需要将数据库元素 X 的取值 v 改变时，undo 日志就会将形如$\langle T,X,v\rangle$的日志记录记到磁盘上，当需要对事务 T 进行恢复，则需要在外存中重新写入 X 的值 v.

若将这个过程移到闪存上，举例来说，如果数据库中存在一个如表 1(a)所示的数据表，当需要将数据表中的 A 值由 v_1 修改为 v_2 时，就要写入一条新的记录，如表 1(b)所示得最后 1 行. 而如果需要将 A

值进行恢复的时候，就要再写入一条其实早已存在于内存中的记录，如表 1(c)所示. 这就可以看出，最后 1 条记录和第 1 条记录是相同的. 也就是说，这其实是存在冗余的. 因此，闪存中通常存在着大量的数据的历史版本，而显式的恢复过程又不断地写入已经存在的数据项. 这既浪费空间又浪费时间.

表 1 undo 日志在闪存数据库中存在的问题

(a) 数据表初始状态

	value	flag
A	v_1	1
B	v_b	1
C	v_c	1
…	…	…

(b) 修改 A 的取值后的表

	value	flag
A	v_1	0
B	v_b	1
C	v_c	1
A	v_2	1
…	…	…

(c) 对 A 值进行恢复后

	value	flag
A	v_1	0
B	v_b	1
C	v_c	1
A	v_2	0
A	v_1	1
…	…	…

同时，我们已经知道，闪存通常采取异地更新，但是因为闪存每次写的单位为页，即使是有所改进的闪存，其一页也通常只能写 4 次. 也就是说，不管一次要写入的数据量多大，至少需要占用闪存中四分之一个页的大小，而通常来说，一个需要恢复的数据项可能并没有这么大. 这样，就更带来了额外的空间浪费.

同时，这些额外的写操作会有较高的时间代价，并且因为一些额外的空间浪费，就会带来一些本不必要的擦除操作，其时间代价更为巨大. 因此，在恢复中所进行这种大量的小的随机重写对闪存的代价也是非常可怕的，这就需要设计新的恢复方法.

2.3 相关工作

随着技术的不断发展，闪存的优势越来越明显，有越来越多的研究关注于如何在基于闪存的数据库中提供更高的性能，其中较有影响力的工作包括 IPL[8]、FlashLogging[9]、Transactional Flash[10] 和 PORCE[11]等.

IPL 彻底改变了闪存数据库中的存储结构，它将闪存上每个块中的页分为两个部分：数据页和日志页. 当对一个块中的数据页进行修改时，为了避免闪存的原地更新带来的巨大代价，IPL 只是将修改以日志的形式保存在其数据所在块的日志页中. 并且在日志区域满时，进行日志记录与数据的合并，来减少存储空间. 这种存储方式因为将对数据库的改变通过日志方式保存，可以间接的对数据库提供恢复. 但是，这需要对现有的数据库进行较大的修改，并不能方便地移植于不同的数据库中.

FlashLogging 提出了一种使用多个性价比高且更适合于日志的访问和存储模式的 USB 设备，来取代 SSD 记录日志. 因为 USB 的存储容量一般来说相对较小，FlashLogging 设计了一种轮转式的阵列组织方式来有效地管理这些分散存放在 USB 设备中的日志记录，并提供恢复. 这种方法需要大量的 USB 设备阵列，并且会需要对多个 USB 设备进行读写，这是非常耗时的，而且随着 SSD 价格的不断下降，USB 设备的价格优势也在渐渐消失，因此，这并不是一个方便的系统搭建模式.

另外，在闪存中，若使用 FTL 层来屏蔽 Flash 的物理特性，则需要维护一个物理地址和逻辑地址的映射表，而将闪存作为嵌入式系统的存储设备时，则因为常常会出现断电的情况，就容易丢失这个映射表. 因此，PORCE 提供了一种在断电之后如何提供物理地址和逻辑地址映射的恢复方法. 而针对基于闪存的文件系统，SAC2006[12] 和 TOS2006[13] 提出了一种如何利用闪存的特性，来提供对基于闪存的文件系统的快速载入和崩溃之后的恢复的方法. 然而，这些方法是针对于文件系统，而不是我们讨论的数据库系统，虽然设计思路上可以有较好的参考，但其性能并不能直接地与我们的设计相互比较.

3 HV-recovery

通过之前的分析，可以发现，在闪存存储设备中，在恢复时，完全没有必要用显式的回滚操作来重新写入数据元素在事务更新前的内容. 考虑到在闪存中数据项新旧版本的同时存在，可以利用旧版本来加快回滚和恢复的过程，而不需要发起更为昂贵的写操作来多次写入一个已经存在的数据项内容. 本文的设计就是考虑最大限度地利用数据项之前的历史版本来进行恢复. 图 1 是 HV-recovery 的一个整体的体系结构图.

在数据库的正常运行阶段，随着事务对数据文

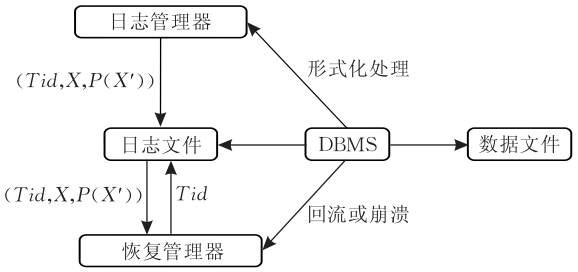


图 1　HV-recovery 体系结构图

件的不断更改，日志管理器同时将形如(Tid, X, $P(X')$)的日志记录存放在日志文件中，当数据库发生崩溃或是要对事务进行回滚时，恢复管理器就根据得到的需要进行恢复的事务的 ID 读取日志文件，获得相关的日志记录，并根据这些日志记录进行恢复. 以下将详细介绍其中各部分的具体实现.

3.1　更新日志

在对日志记录进行管理时，HV-recovery 使用了一个版本列表 version_list 来保存日志记录，其结构如表 2 所示. 作为日志记录，其中存储的是被修改的各数据库元素的历史版本的地址信息、引起该数据库元素更改的事务的标识以及该数据项的旧值.

表 2　version_list 结构图

Tid	元素	PreAddress	PreValue
T_1	X	$P(X')$	X'
T_2	A	$P(A')$	A'
T_4	D	NULL	NULL
T_4	B	$P(B')$	B'
T_4	B	Delete	NULL
T_6	C	$P(C')$	C'
T_2	Y	$P(Y')$	Y'
T_1	X	$P(X'')$	X''
T_1	*commit*	NULL	NULL
…	…	…	…

当需要数据库系统对数据项进行更新时，在将新的数据版本写入新地址的同时，日志管理器会将这个数据项的旧值、旧地址以及该事务标识存入到 version_list 中. 并将其作为日志记录保存在永久性存储器中. 同时，HV-recovery 的日志记录类似于 undo 日志，必须遵守两条规则.

规则 A.　如果事务改变了数据库元素，则日志记录必须在数据库元素的新值写到二级存储器前写出.

规则 B.　如果事务提交，则其事务提交日志记录操作必须在事务改变的所有数据库元素已写到二级存储器之后再执行，但应尽快.

这样，在恢复中，只要对于在日志记录中显示为未提交的事务，对在日志记录中所保存的该事务所做的所有修改进行还原，就可以保证数据库对于事务所要求的 ACID 特性.

另外，在 HV-recovery 中，若同一个事务对某一个它已更新过的数据项又有新的更新，就增加一条新的日志记录，保存新的更新操作. 如表 2 的第 1 条记录和第 8 条记录所示.

而如果是不同事务对同一数据项进行修改，在满足数据库对于并发设计的要求的前提下，对日志记录部分来说，也是产生一条新的日志记录在 version_list 中. 如表 2 中，第 2 条记录和第 7 条记录所示，T_2 修改了 A 之后又修改了数据项 Y，那日志管理器就将这两次修改作为两条不同的日志记录来存储.

若数据库插入一个新的数据库元素，则与更改日志记录类似，产生一条新的日志记录，只是该日志记录的旧值和旧地址项被设置为空，以识别为插入操作. 如表 2 中第 3 条记录所示.

若数据库删除一个数据项，在写入原有的日志记录的同时，再增加一条日志记录，使用相同的事务标识和数据项，但是将其旧版本地址设置为一个定义了的删除标识. 如表 2 中第 4 条记录和第 5 条记录所示，T_4 删除了一个数据库元素 B，则为 T_4 和 B 增加两条新的日志记录，并把后一条日志记录的历史版本的值设置为 delete 标识，旧值设为空.

3.2　事务提交日志

在 HV-recovery 中，每当有事务提交，就在日志记录文件，也就是 version_list 中对该事务添加一个新的提交记录. 具体来说，就是 HV-recovery 为每个事务设置了一个 *commit* 元素，当某个事务提交时，就为该事务增加一条日志记录，在这个日志记录中，将该事务的 *commit* 元素的地址置为空. 如表 2 中第 9 条记录所示，当 T_1 事务提交，日志管理器就对 T_1 事务设置 *commit* 元素，并将该日志记录的旧版本地址字段设为空.

注意，因为本文的日志记录必须满足规则 B，也就是说，当事务提交日志记录到达二级存储器的时候，该事务所改变的所有数据库元素已经写到二级存储器上了. 因此，此时数据库已经提交了该事务的所有更新操作. 相反，如果事务提交日志记录未到达二级存储器，则在恢复时，不管这个事务的修改在数据库中完成了多少，这个事务所做的所有操作都将被还原，从而保证事务的原子性.

另外，日志文件中存在的已经提交的日志记录会增加日志文件的长度，同时，会使得对日志文件的访问变成随机模式. 但是，正如之前所介绍的，因为闪存设备的随机读和连续读的访问时间的差异不明

显,所以这些日志记录的存在对于其它事务的恢复效率的影响是非常微小的,几乎可以不必考虑.

不过,这种日志记录长久保存是以大量的闪存存储空间为代价的,考虑到当前闪存存储设备的价格还不是很低廉,这会使得应用系统的成本价格提高,所以在第 4 节中,本文会提供一个进一步改进的方案.

3.3 恢复过程

当系统发生崩溃或者事务需要进行回滚时,由于 HV-recovery 对日志记录和数据更新的提交顺序满足规则 A 和规则 B,所以,只需要对在日志记录中体现为尚未提交的事务进行恢复.

首先,像其它恢复方式一样,先读入在数据库的二级永久性存储器中需要恢复事务的日志记录.对于同一个事务修改的同一个数据项的所有记录,选择所有记录中的第一条记录.因为日志记录是顺序添加的.而一个事务对某一数据项不断地更改,就不断地在后面添加新的日志记录,这就保证了其第一次保存的历史版本的地址恰好就是在该事务修改之前的数据项的值.

然后根据这些日志记录,恢复管理器读出需要恢复的各数据项的历史版本的地址,从纸质中取出其所存的数据,判断是否与日志记录中存的旧值相同,若相同,将已写入新更新数据内容的地址标识为无效,将原地址标识为有效,并将原地址赋给上层索引结构,从而完成恢复;若不同,则只有重新写入.

4 HV-recovery 方法的改进

为了进一步提高 HV-recovery 的性能表现,在本节中,针对 HV-recovery 中还存在的一些问题提供了一些改进的方法.

4.1 设立检查点

为了减少数据文件和日志文件对存储空间的浪费,本文采取了一种周期性设立检查点的措施,来有效地提高空间利用率.

4.1.1 对于日志文件的操作

首先,可以看出,HV-recovery 中的日志记录的更新是非常频繁的,随着事务的不断进行,需要不断地向日志文件中添加新的日志记录,而随着事务的不断提交,又使得大量的日志记录变为无效.而在一般情况下,事务的回滚率通常不会太大,也就是说,其实在日志文件中存在着大量无效的日志记录,而这些日志记录的存在,增加了日志文件的长度,也占用了过多的闪存存储空间.

因此,在检查点中采取一种最简单的转移操作,将日志文件中尚有效的日志记录进行转移并进行整合.也就是说,在设立的检查点中,先找到一个干净的块,然后逐条检查每条日志记录是否有效,在日志记录中选出尚有效的记录,写入到新的空闲块,当对某一旧日志块上的所有日志记录都检查过一遍后,就对该旧块进行擦除.通过之前的介绍,可以看出,实际需要转移的日志记录相对于大量的已经提交的事务的日志记录而言,其数量是相当小的,这样其转移的代价也是可以接受的.

4.1.2 对于数据文件的操作

由于我们在闪存中采取异地更新,因此,为了显式地使用历史版本的数据,在实现时,在存储设备看来,我们将更新操作改为了插入,而删除操作只是记录了日志,并没有将历史数据删除,或标识为无效,这样虽然防止了我们将需要的历史版本的数据进行回收,但同时造成了系统中有过多的历史数据,存储空间大量浪费.

因此,在检查点时,我们会在对日志记录进行扫描的同时,将无效的日志记录中显示为应该被删除或替代掉的数据标识为无效,以便让垃圾回收机制对空间进行回收管理,提高空间利用率.

4.1.3 检查点时间间隔设置

检查点的间隔时间的确定,是与闪存中日志文件的大小以及闪存的总存储空间有关的.间隔时间太短会因为过多的擦除操作而浪费时间,并缩短闪存的寿命,而间隔时间过长又会浪费存储空间.因此,可以根据闪存的总存储空间设计固定的可接受的日志文件的大小.当到达某个阈值的时候就设立检查点开始进行转移操作.

同时,类似于 undo 日志的检查点,不但可以设立静态的检查点,也可以设置动态的检查点.在检查点的开始阶段保存正在活跃的事务的 ID,这样,就可以不必等到所有事务都提交完毕后,再设立检查点,对日志文件进行转移.并且也可以在数据库负载量较小的时候进行检查点的日志记录转移操作,从而进一步减少系统负担.

4.2 混合式存储系统

另外,也可以看出,HV-recovery 中对日志记录的主要操作就是一些小的追加写操作和一些擦除操作.而由之前对闪存的硬件特性的介绍可以得知,这些操作对于闪存而言是非常昂贵的.因此,可以看出,日志记录其本身的特点是不适用于闪存的.

而一般来说,基于闪存的数据库是指将大量的对其操作较多的数据文件保存在闪存中,以利用其

优越的读写速度来提供更好的数据库性能.而当前的数据库一般都支持将日志文件和数据文件分开存储,因此,可以考虑在存储时使用混合式系统,如图 2 所示.将数据记录存放在闪存上,同时将并不是非常适用于闪存的日志记录存放在磁盘中.

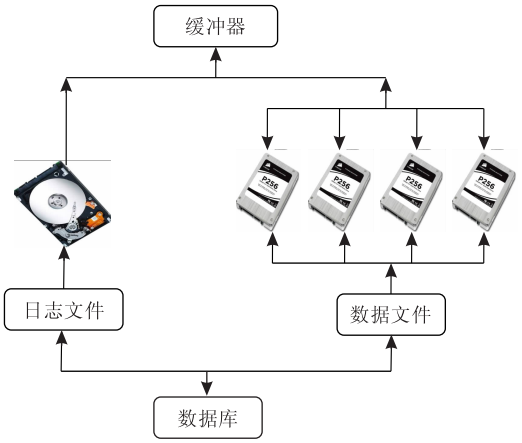


图 2　混合式存储系统结构图

通过这样的设计,就可以在不增加系统恢复算法复杂度的同时,节约大量的日志记录所占用的闪存空间,为数据库系统服务.降低了数据库系统搭建的成本的同时,从整体上并没有影响数据库系统的性能表现.

5　实验结果及分析

本章通过对 HV-recovery 在闪存上和磁盘上的实际对比实验,从恢复时的写操作数、恢复时间等方面来证明 HV-recovery 的优越性.

5.1　实验环境

本文设计了两个实验平台,其中一个的存储设备配置了 SSD,是 80GB 的 Intel SSDSA2MH080G1GC,另一个配置的是磁盘,是 250GB 7200rpm ST3250310AS,其中有 8MB 的缓存.除此之外,两个实验平台具有相同的配置,SSD 和磁盘都是通过 SATA 接口接入.CPU 是 Intel(R) Core(TM) 2 Duo CPU E8300 @ 2.83GHz,物理内存为 2GB,操作系统是 Windows XP Professional 2002 Service Park 2.

5.2　与相关工作的对比与分析

之前的介绍中提到,IPL 主要是给出一种新型的存储模式,以提供性能较高的对数据库的操作.它所提出的恢复技术主要是针对这种新的存储方式的一种扩展,因此,IPL 不能像 HV-recovery 一样,方便地扩展到现有的大量数据库中.同时,IPL 的存储方式是针对原始的闪存存储器的,而不是现在被简单广泛地应用在 SSD 上,因此,IPL 具有相当大的局限性.

同时,因为 IPL 的论文中并没有给出针对其恢复性能的实验结果,并且,其设计思想的实现必须在原始的闪存存储器上,而论文中也没有对其实现细节进行详细的阐述,这就对我们重现其工作带来了较大的困难,难以提供定量的对比结果.

另外,FlashLogging 给出在 TPCC 执行过程中突然崩溃需要扫描日志记录的时间,大约是基于磁盘的 2/3 左右,而 HV-recovery 的恢复时间是基于磁盘的 1/8 左右(在 5.4 中会详细阐述).

当然,两种设计的实验环境和对比细节不完全相同,把实验结果直接进行比较不是很具有说服力.但是,由于 FlashLogging 要求搭建 USB 阵列,我们在短时间内较难重现,因此,我们在本文中难以给出直接的量化比较.但是,从单纯的恢复时间的比较,我们至少可以相信,HV-recovery 的性能表现并不会比 FlashLogging 的表现差.

5.3　恢复时的写操作数

TPC 提供了一系列系统性能的压力测试标准,其中的 TPCC 通过规定数据库原始数据生成以及查询负载的相关指导标准来模拟了 OLTP 的处理场景,是数据库系统事务处理性能的标准测试集.

TPCC 规定了在 OLTP 中典型的 5 种事务,包括 New-Order、Payment、Order-Status、Delivery 以及 Stock-Level,在下面的分析计算中,对这些事务类型分别简称为 T_1、T_2、T_3、T_4、T_5.并且,TPCC 模拟实际情况,设定了这 5 种事务各自所占的比例,分别为 45%、43%、4%、4%、4%,不妨将这些值用 P_1、P_2、P_3、P_4、P_5 表示.TPCC 还根据实际情形为每一种事务定义了一系列的插入、删除和更新操作,不妨把每种事务需要进行的操作数记为 N_1、N_2、N_3、N_4、N_5,如表 3 所示.

表 3　TPCC 关于 5 类事务的规定

事务	类型	所占比例/%	操作数
New-Order	T_1	$P_1=45$	N_1
Payment	T_2	$P_2=43$	N_2
Order-Status	T_3	$P_3=4$	N_3
Delivery	T_4	$P_4=4$	N_4
Stock-Level	T_5	$P_5=4$	N_5

TPCC 模拟了大量用户同时对系统进行并发访问的模式,此处设存在的用户并发数为 N_{user}.同时,容易理解,在任一时刻,正在并发执行的事务都完成了其中的一部分,也就是说,每个事务都有一个自己

的完成率，它是一个从 0～100％之间的一个随机数，记为 C.

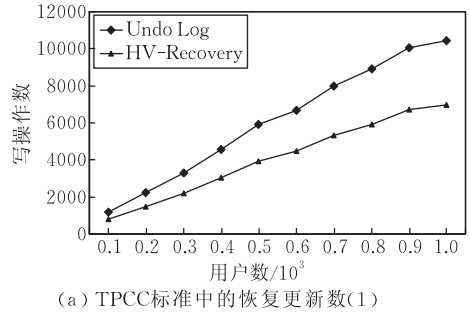
因此，可以知道，对一次有 N_{user} 个并发的 TPCC 测试而言，任一时刻系统发生崩溃，其需要恢复的数据量 N_{recovery} 为

$$N_{\text{recovery}}=\sum_{i=1}^{5}N_i\times\left(\sum_{j=0}^{N_{\text{user}}}T_j\times C_j\right),$$

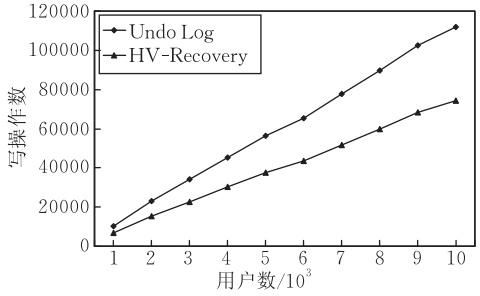
其中保证对每一种事务的相互比例满足 TPCC 的要求，也就是说，$T_1:T_2:T_3:T_4:T_5=P_1:P_2:P_3:P_4:P_5$.

通过之前的介绍可以看出，在恢复过程中，对于每一个需要进行恢复的数据项，HV-recovery 都可以比传统的恢复方式减少一次写操作，也就是说，HV-recovery 只需要 $2N_{\text{recovery}}$ 个写操作和 N_{recovery} 个读操作就可以完成恢复，而传统的恢复方式至少需要 $3N_{\text{recovery}}$ 个写操作.

实验结果如图 3(a)、(b)所示，可以发现，随着并发用户数的不断增加，与传统的 undo 日志相比，在数据库恢复阶段，HV-recovery 可以大量地减少写操作. 一般而言，在并发用户数从 100 增加到 10000 时，HV-recovery 可以减少大约有 400～37000 次写操作. 在减少写操作的同时，也节约了大量的闪存空间，提高了闪存的空间利用率.



(a) TPCC标准中的恢复更新数(1)



(b) TPCC标准中的恢复更新数(2)

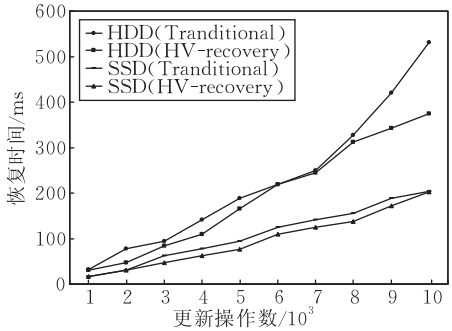
图 3 在 TPCC 标准下的实验结果

同时，因为写操作的时间代价比较大，从理论上分析，HV-recovery 可以在恢复时节省大量的时间. 因此，我们将 HV-recovery 实现到现有的数据库中，用实验结果来验证其可以节省大量的恢复时间.

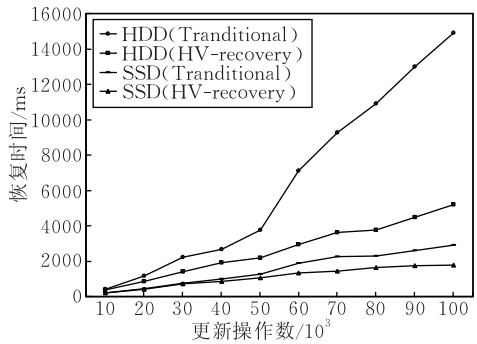
5.4 在 Berkeley DB 中实现 HV-recovery

为了方便地将 HV-recovery 在现有的数据库中进行实现，本文选择的是开源的 Oracle Berkeley DB 数据库，编程语言为 C 语言，编译环境是 Microsoft Visual Studio 2005，硬件环境如 5.1 节所述.

在实验中，在开始后不断地对数据库中的内容进行各种操作，然后，再显式地对事务回滚，即进行恢复操作，并记录恢复所耗费的时间，其结果如图 4 所示. 由实验结果可以看出，HV-recovery 相比于传统的基于磁盘的数据库有明显的优越性. 能够大幅度的减少恢复时间. 并且随着数据量的不断增加，这种优越性也越来越明显. 在较小的数据量时，如图 4(a)所示，使用传统数据库恢复技术所用的恢复时间平均是使用 HV-recovery 进行恢复的 2～3 倍，而数据量不断增大时，HV-recovery 的优势也成倍增加. 如图 4(b)所示，最好情况下，基于磁盘上的传统数据库的恢复时间是 HV-recovery 恢复时间的 8.3 倍，充分体现了 HV-recovery 的优越性.



(a) Berkeley DB中的恢复时间(1)



(b) Berkeley DB中的恢复时间(2)

图 4 在 Berkeley DB 中的实验结果

而且，与将传统的数据库直接移到 SSD 上的情况相比，HV-recovery 也体现了较大的优势. 一般而言，在 SSD 上，HV-recovery 要比 Berkeley DB 中传统的 undo 日志的方法要提高 20％，而随着数据量的增大，这种优势也体现的更为明显，如图 4(b)所示，某些情况下，HV-recovery 要比 Berkeley DB 中

传统的 undo 日志的方法要提高 38%，能很好地体现 HV-recovery 的优越性.

6 结　论

闪存即将取代磁盘成为下一代主流的二级存储器，但由于其特有的物理特性，目前基于磁盘设计的恢复技术不能充分地利用闪存的优越性. 为此，本文提出了一种新颖的具有较高性能的恢复方式 HV-recovery.

HV-recovery 对闪存中天然存在的数据的历史版本使用 version_list 结构加以管理和利用，提供高效的恢复. 通过周期性设立检查点，减小无效日志记录的长度，节约闪存空间. 引入混合式存储系统，将日志记录单独存放在磁盘上，以便对闪存数据库的恢复性能进一步提高. 同时也保证了算法具有在数据库正常运行时有较小的开支、算法有比较强的可靠性、系统失败后恢复速度快和日志文件的空间需求较小等优势.

通过针对 TPCC 的分析及和开源数据库 Oracle Berkeley DB 的对比实验看出，HV-recovery 比传统数据库的恢复时的写操作数可以减少接近一半，其恢复时间与传统数据库相比，能缩短到原来的大约 1/8，与在 SSD 上的传统数据库相比，也可以缩短 40%，充分显示了本算法的优越性.

参　考　文　献

[1] Gray Jim, Fitzgeral Bob. Flash disk opportunity for server applications. ACM Queue, 2008, 6(4): 18-23

[2] Lee S W, Moon B, Park C, Kim J M, Kim S W. A case for flash memory SSD in enterprise database applications//Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, Canada, 2008: 1075-1086

[3] Lee Sang-Won, Moon Bongki, Park Chanik. Advances in flash memory SSD technology for enterprise database applications//Proceedings of the 35th SIGMOD International Conference on Management of Data. Providence, USA, 2009: 863-870

[4] Kim Yi-Reun, Whang Kyu-Young, Song Il-Yeol. Page-differential logging: An efficient and DBMS-independent approach for storing data into flash memory//Proceedings of the 2010 International Conference on Management of Data. Indianapolis, USA, 2010: 363-374

[5] Haerder T, Reuter A. Principles of transaction-oriented database recovery. ACM Computing Surveys, 1983, 15(4): 287-317

[6] Reuter A. Performance analysis of recovery techniques. ACM Transactions on Database Systems, 1984, 9(4): 526-559

[7] Garcia-Molina Hector, Ullman Jeffrey D, Widom Jennifer. Database System Implementation. USA: Prentice Hall, 1999

[8] Lee Sang-Won, Moon Bongki. Design of flash-based DBMS: An in-page logging approach//Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. Beijing, China, 2007: 55-66

[9] Chen Shimin. FlashLogging: Exploiting flash devices for synchronous logging performance//Proceedings of the 35th SIGMOD International Conference on Management of Data. Providence, USA, 2009: 73-86

[10] Prabhakaran Vijayan, Rodeheffer Thomas L, Zhou Lidong. Transactional flash//Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation. San Diego, USA, 2008: 147-160

[11] Chung Tae-Sun, Lee Myungho, Ryu Yeonseung, Lee Kangsun. PORCE: An efficient power off recovery scheme for flash memory. Journal of Systems Architecture: the EUROMICRO Journal, 2008, 54(10): 935-943

[12] Wu Chin-Hsien, Kuo Tei-Wei, Chang Li-Pin. Efficient initialization and crash recovery for log-based file systems over flash memory//Proceedings of the 2006 ACM Symposium on Applied Computing. Dijon, France, 2006: 896-900

[13] Wu Chin-Hsien, Kuo Tei-Wei, Chang Li-Pin. The design of efficient initialization and crash recovery for log-based file systems over flash memory. ACM Transactions on Storage, 2006, 2(4): 449-467

LU Ze-Ping, born in 1985, M. S. candidate. Her current research interests include recovery of Flash-based database systems.

MENG Xiao-Feng, born in 1964, Ph. D., professor, Ph. D. supervisor. His research interests include web data management, native XML databases, mobile data management, etc.

ZHOU Da, born in 1980, Ph. D. candidate. His current research interests include indexing, query processing and transaction processing of flash-based database systems.

Background

Due to its superiority such as low access latency, low energy consumption, light weight, and shock resistance, the success of flash memory as a storage alternative for mobile computing devices has been steadily expanded into personal computer and enterprise server markets with ever increasing capacity of its storage. However, since flash memory exhibits its poor performance for small-to-moderate sized writes requested in a random order, existing database systems may not be able to take full advantage of flash memory without elaborate flash-aware data structures and algorithms.

We consider the recovery technique for flash based database systems. Now there is few researches on this issue, and the works has been published is either use a special storage structure or based on devices which are difficult to rebuild.

Our research group focuses on the design and implementation for flash-based database systems. We mainly solve the degradation of performance when we transfer the traditional database to flash memory without any modification. We have published several papers about index and buffer management for flash-based databases on internal and external conferences. And this work is the first one about recovery in our group. This paper is used for improve the recovery performance for flash-based databases.

In this paper, we proposed a new recovery method called HV-recovery to provide recovery in flash based database systems without too many changes on current databases. HV-recovery makes use of the history versions of data which is naturally emerged in flash memory due to the out-of-place update. So it just needs to change the recovery component and logging component of databases, and could leave the others as it is. So HV-recovery is convenient to transfer to almost all the commerce databases. Meanwhile, HV-recovery is also effective. Experimental results on Oracle Berkeley DB show that our HV-recovery outperforms traditional recovery in 2×～8×. The results demonstrate the high efficiency of our method.

This research was partially supported by the grants from the National Natural Science Foundation of China under grant Nos. 60833005, 60573091; National High Technology Research and Development Program (863 Program) of China (Nos. 2007AA01Z155, 2009AA011904).