基于相变存储器和闪存的数据库事务恢复模型

范玉雷 孟小峰
(中国人民大学信息学院 北京 100872)

摘 要 随着闪存容量不断增大、价格不断下降,闪存在实际存储系统上得到了越来越广泛的应用.但是,闪存的页级读写、异位更新、有限寿命等阻碍了闪存数据库系统的性能提升,尤其是事务恢复.闪存的异位更新使得影子页技术可以很好地支持闪存数据库事务恢复,同时也给闪存数据库带来新挑战,如事务管理、缓冲区管理.相变存储器凭借其比闪存更高的读写速度、更小的读写粒度、更长的寿命成为了下一代主流存储技术,所以相变存储器可以用于解决在闪存数据库中使用影子页技术所产生的事务管理和缓冲区管理问题.该文基于相变存储器和闪存混合式存储提出一种全新的数据库事务恢复模型——SPFP.该模型充分利用相变存储器的特性完成事务管理.为支持非强制缓冲区管理,基于 SPFP 提出了一种优化的数据库事务恢复模型——SPFLP,利用相变存储器记录更多事务信息.实验结果表明,相较于全闪存存储的数据库系统,SPFLP 大大提高了基于混合存储的数据库事务处理性能.

关键词 闪存;相变存储器;数据库;事务恢复
中图法分类号 TP311 **DOI 号** 10.3724/SP.J.1016.2013.01582

Transaction Recovery Model of Databases Based on PCM and Flash Memory

FAN Yu-Lei MENG Xiao-Feng
(*School of Information, Renmin University of China, Beijing* 100872)

Abstract The capacity and the cost of flash memory are being advanced and reduced continuously, so they are more and more widely used in the actual storage systems. But the unique characteristics of flash memory, such as page-level read/write, out-of-place update, limited lifetime, hinder promoting the performance of flash-based database systems, especially transaction recovery. Because of out-of-place update of flash memory, shadow paging technology can be adopted to support transaction recovery in flash-based DBMS, but new challenges are brought, such as transaction management and buffer management. Because of the unique characteristics of Phase Change Memory (PCM) such as higher read/write speed, smaller grain size of read/write, more durable than flash memory, PCM, as one of the most promising next-generation memory technologies, can be used in transaction management and buffer management when shadow paging technology has been adopted in flash-based DBMS. Based on hybrid storage of PCM and flash memory, this paper presents a new recovery model for databases, SPFP, which exerts advantages of PCM for managing transaction. For supporting no-force buffer management, we propose an optimized transaction recovery model, SPFLP, which records more transactions' information in PCM. Our preliminary experiments show that there is a bigger improvement of SPFP and SPFLP in transaction processing performance of databases based on PCM and flash memory in comparison with these based on flash memory.

Keywords flash memory; Phase Change Memory (PCM); atabase; transaction recovery

收稿日期:2013-03-15;最终修改稿收到日期:2013-06-03.本课题得到国家自然科学基金(61070055,91024032,91124001)、核高基重大专项基金(2012AA010701,2013AA013204)和中国人民大学科学研究基金(11XNL010)资助.**范玉雷**,男,1984 年生,博士研究生,主要研究方向为闪存数据库存储管理、查询处理和查询优化、索引和恢复技术研究.E-mail:fyl815@ruc.edu.cn.**孟小峰**,男,1964 年生,博士,教授,博士生导师,主要研究领域为网络数据管理、云数据管理、移动数据管理、社会计算、闪存数据库系统、隐私保护.

1 引　言

随着闪存技术的飞速发展，闪存的容量在不断增大，闪存的单位价格也在不断降低，因此闪存在存储系统中得到了广泛应用，小到传感器，大到大型服务器，都充分发挥了其小巧轻便、抗震、耐高/低温、耗电量小、读写速度快等特性[1]. 闪存具有完全不同于磁盘的物理特性，如非对称读写、页级读写、写前擦除、异位更新、有限寿命等，所以学术界和工业界在基于闪存的数据库系统的研究方面进行了大胆尝试并取得了突出的科技成果.

相变存储器(Phase Change Memory，PCM)是非易失随机访问存储器(Non-Volatile Random Access Memory，NVRAM)中的一种，同时还是具有磁盘和内存优势特性的下一代主流存储技术[2]. 相比于内存和闪存，相变存储器提供了很多吸引人的适合于数据库事务处理的新特性. 相比于内存，相变存储器具有与内存相近的读速度，同时还具有非易失性. 相比于现有的闪存，相变存储器具有高出两个数量级以上的读写速度，同时还具有细粒度的原位更新特性，并且相变存储器不具有闪存的写前擦除特性. 所以研究者将要面临新的问题“如何修改数据库使其充分利用新硬件相变存储器的优势特性”[3]. 因此本文充分考虑相变存储器，重新审视闪存数据库的事务处理模块.

在数据库管理系统中，事务恢复是数据库事务处理的一个重要模块. 写前日志技术[4]和影子页技术[5]是两种经典的事务恢复技术. 写前日志技术是基于磁盘的原位更新设计的，原位更新使得原有旧版本数据被覆盖，故通过记录日志来记录更新操作，在恢复过程中通过读取日志进行相应的 Redo/Undo操作进而获取最准确的数据. 写前日志技术在记录数据更新的同时，还需要一些特殊的日志用来记录事务状态、检查点等信息，用以完成事务恢复. 由于闪存存储器多采用异位更新模式，所以写前日志技术不适用于基于闪存的数据库系统. 影子页技术是基于异位更新设计的，故影子页技术可以很好地用以支持闪存数据库的事务恢复，但同时影子页技术对闪存数据库也提出了两个新的挑战——事务管理和缓冲区管理. 首先，在事务执行和恢复过程中需要进行有效的事务元信息管理. 其次，像磁盘数据库一样，闪存数据库系统也需要支持非强制的缓冲区管理策略，进而提高整个系统的性能.

相变存储器具有内存和闪存的优势特性，如按位修改、非易失等，所以本文利用相变存储器解决影子页技术应用在闪存数据库系统中所带来的事务管理和缓冲区管理问题. 由于相变存储器允许按位修改、具有与内存相当的读速度、具有比闪存高两个数量级的读写速度，所以可以使用相变存储器对事务进行管理，包括事务 ID 和事务状态等信息. 同时为了支持非强制的缓冲区管理策略，需要对事务进行的操作进行更详细的记录和维护，比如事务操作类型和事务操作内容等.

本文针对基于相变存储器和闪存的数据库系统设计了全新的事务恢复模型. 总体来说，本文的主要贡献如下：

(1) 提出了基于相变存储器和闪存的数据库事务恢复模型 SPFP，并阐述了在 SPFP 模型下的正常事务处理方式；

(2) 设计了适用于 SPFP 模型下的数据库故障恢复方法和闪存空间回收方法；

(3) 提出了支持非强制缓冲区管理的数据库事务恢复模型 SPFLP，并阐述了在 SPFLP 模型下的正常事务处理方式；

(4) 设计了适用于 SPFLP 模型下的数据库故障恢复方法.

本文第 2 节简单介绍闪存、相变存储器以及相关工作；第 3 节提出基于相变存储器和闪存的数据库事务恢复模型 SPFP，并阐述事务正常处理方式、数据库故障恢复和闪存空间回收方法；在第 4 节，本文提出基于 SPFP 优化的数据库事务恢复模型 SPFLP，并详细描述事务正常处理方式、数据库故障恢复方法；第 5 节通过实验对比分析阐述 SPFP 和 SPFLP 的性能优势；在第 6 节作简单总结.

2 研究背景及相关工作

由于闪存和相变存储器都各自具有一些适合于数据库的优良特性，所以需要结合两者的优势充分提高数据库系统的性能.

2.1 闪存和相变存储器

和磁盘一样，闪存和相变存储器都是非易失存储器. 如图 1 所示，闪存设备由一组闪存芯片组成，每个闪存芯片内部按照块进行组织，块是闪存擦除操作的基本单元. 块是由若干页组成，页是读写操作的最小单元. 一个闪存页分为数据区和空闲区，数据区主要存储用户数据，空闲区主要存储校验码和逻

辑页地址等元信息.典型的数据页大小为 2 KB+64 KB,其中 2K 为数据区,64B 为空闲区.NAND 闪存常用于存储数据,根据存储密度又可以把 NAND 闪存分为 SLC NAND 闪存和 MLC NAND 闪存[6].由于 MLC NAND 闪存的应用广泛性,本文意在研究基于 MLC NAND 闪存的数据库系统.

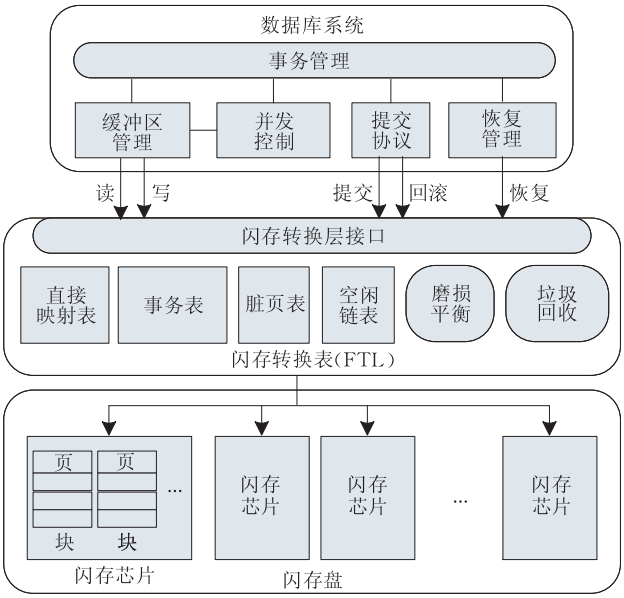

图 1 闪存数据库系统

不同于磁盘,闪存无机械组件,所以闪存具有较高的读写性能[7],并且单个请求响应延迟主要取决于数据传输的总量.由于闪存具有写前擦除等特性,异位更新常被实现在闪存转换层(Flash Translation Layer,FTL)中,用于解决写前擦除的约束[8].如图 1 所示,闪存转换层是承接上层应用的关键组件.闪存转换层为上层提供读写接口和事务操作接口(提交、回滚和恢复).

如图 2 所示,闪存转换层在内存维护一个直接映射表(Direct Mapping Table,DMT),用于记录逻辑地址到物理地址的映射;每个闪存物理页的空闲区内的逻辑地址形成了一个反向映射表(Inverse Mapping Table,IMT),用于在系统启动时在内存重建直接映射表.直接映射表和反向映射表对于异位更新的实现是至关重要的.

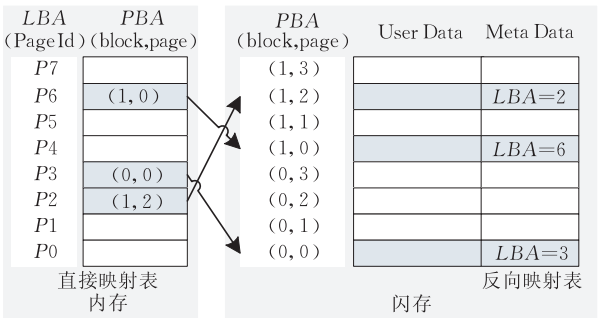

图 2 FTL 映射表

异位更新策略产生了大量的无效数据页,故在闪存转换层维护了一个垃圾回收模块,定期的或者不定期的回收无效数据页.由于闪存块具有有限的擦除次数,单个闪存块的损坏会导致整个闪存芯片不可用,所以需要在闪存转换层中维护一个磨损均衡模块,用以实现整个闪存的均匀使用.根据地址映射的粒度可以把闪存转换层分为 4 类:页级映射、块级映射、块页混合映射和其它[9].

相变存储器采用了与闪存完全不同的材料,采用材料的晶体和非晶体两种状态来区分 1 和 0.内存、磁盘、闪存和相变存储器的硬件性能如表 1 所示[3].除了非易失和高密度之外,相变存储器还具有如下特征[1-2,10]:(1)细粒度的操作.与闪存相比,它没有写前擦除和基于页的访问约束,允许按位修改和访问,所以可以像内存一样支持小粒度的原位更新;(2)非对称的读写速度.写延迟大概是读延迟的 20 倍;(3)有限寿命.但比闪存寿命要长,可支持 $10^6 \sim 10^8$ 次写操作.

表 1 常用硬件性能对比[3]

参数	密度	读延迟(粒度)	写延迟(粒度)	持久性
内存	1X	20～50 ns(64 B)	20～50 ns(64 B)	N/A
PCM	2～4X	～50 ns(64 B)	～1 μs(64 B)	$10^6 \sim 10^8$
闪存	4X	～25 μs(4 KB)	～500 μs(4 KB)	$10^4 \sim 10^5$
磁盘	N/A	～5 ms(512 B)	～5 ms(512 B)	∞

2.2 相关工作

对于基于闪存存储器的数据库系统,事务恢复方面的工作较少.安士通等提出了基于影子页技术的事务提交策略[11],充分利用了 SLC NAND 闪存独有的部分页写特性.在闪存页的空闲区维护操作该数据页的事务元信息,当事务状态发生变化(事务由运行状态变为提交状态或者回滚状态),只需要更新该页的空闲区,而不需要新产生闪存页,因为 SLC NAND 闪存页支持多次写操作.但是 SLC NAND 闪存相比于 MLC NAND 闪存容量较低且单位容量价格较高,使得 SLC NAND 闪存应用范围较小.本文主要针对 MLC NAND 闪存进行数据库恢复研究.需要注意的是 MLC NAND 闪存不具有部分页写特性.在文献[12-13]中,卢泽萍等人设计了基于异位更新优化的日志结构,只记录更新前后的数据页地址,大大减小了日志存储空间.在此基础之上,作者优化了事务恢复策略,利用日志记录的新旧版本数据页地址可以进行快速恢复,同时通过

同一事务日志之间的链接跳读日志减少长事务的恢复时间.但是日志结构的小粒度写并不适合于页级读写粒度的闪存设备.

研究人员开始重新审视基于相变存储器的系统的设计和优化.综合文献[2,14],有3种可用的基于相变存储器的层次架构:(1)相变存储器为主存的辅助存储器,外存为磁盘[2];(2)相变存储器作为主存,磁盘为二级存储设备[2];(3)相变存储器和闪存同时做二级存储设备[14].基于相变存储器存储性能考虑,使用相变存储器作为辅助存储器可能更实用.第一个原因是相变存储器具有持久性的限制,其写入延迟还稍高于DRAM.其次,在未来几年内,相变存储器性能赶不上DRAM性能.基于第1种架构,Gao等人[15]提出一种利用相变存储器减少日志代价的方法,并且在该文中仍然使用相变存储器进行数据页的缓冲.但是相变存储器的写速度距离内存的写速度仍然有两个数量级的差别,并且随着内存技术发展,内存容量也在不断增大,所以尽量充分利用内存缓存数据,使用相变存储器作为永久存储来提高整体性能.Lee等人[16]首先在2007年提出了一种基于页内日志的闪存数据库系统,即在闪存块内分为两个区——数据区和日志区.数据区用以存储用户数据页.日志区只记录当前块内数据页的更新.内存有数据缓冲区和日志缓冲区分别对应数据区和日志区.但是为了尽可能降低数据丢失,日志缓冲区的日志页要小于数据缓冲区的数据页.由于闪存只支持页级读写,日志页的读写浪费了闪存空间,所以Lee等人[14]提出了一种利用相变存储器加速页内日志的方式,数据区更新产生的日志放到相变存储器上,因为相变存储器支持小粒度的读写操作.本文针对第3种存储结构展开研究.

3　基于影子页的数据库恢复模型SPFP

利用相变存储器维护事务元信息,在闪存数据库中实现影子页恢复技术.

3.1　SPFP模型框架

基于影子页的数据库恢复模型SPFP的基本思想:利用影子页跟踪用户对闪存数据页所进行的更新操作,并使用相变存储器记录事务最终状态.如图3所示,闪存数据页的空闲区维护3个信息:该闪存数据页对应的逻辑地址、指向该闪存数据页前一个版本的指针和产生该闪存数据页的事务ID.事务ID是产生该闪存数据页的唯一标识.空闲区信息主要用于完成事务恢复和垃圾回收.MLC NAND闪存设备没有部分页写特性(SLC NAND闪存设备的独有特性),但是为了完成事务正常处理和恢复必须要进行事务管理,即追踪并记录事务执行的状态变化(正在执行/提交/回滚).由于相变存储器允许进行细粒度的访问和原位更新操作,所以本文借助相变存储去实现高效的数据库恢复模型.在事务回滚或者系统启动时,相变存储器的数据和闪存数据页空闲区的元数据可以用来取消事务所进行的更新和系统恢复.

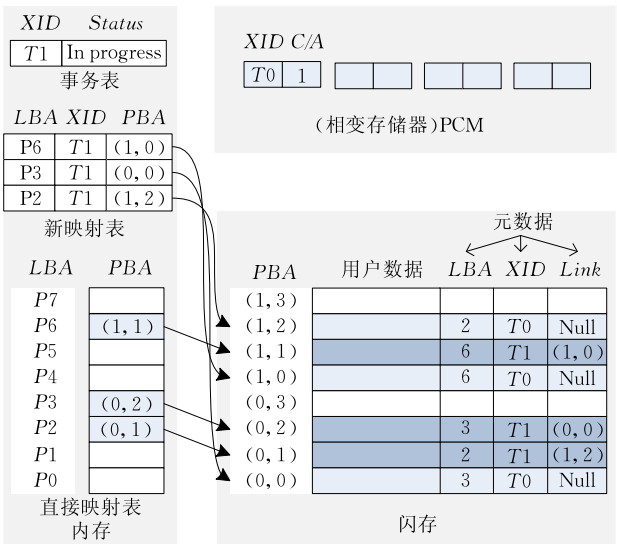

图3　基于影子页的数据库恢复模型SPFP

基于磁盘的数据库管理系统,缓冲区缓存经常访问的数据库磁盘页面.缓冲区管理策略对事务处理能力有巨大影响.在本部分,缓冲区管理策略采用Steal和强制的缓冲区管理策略.换句话说,事务提交之前允许数据页写到闪存上,并且只有事务更新的所有数据页都刷出到闪存之后才可以提交事务.但是对于缓冲区管理,我们需要做一些假设.首先,采用页级并发控制协议处理更新冲突.其次,采用写回缓冲区策略:直到事务提交或者数据缓冲页需要从缓冲区换出的时候,才在闪存上为更新操作真正创建影子页.

内存数据结构有直接映射表、新映射表(New Mapping Table)和事务表(Transaction Table),如图3所示.直接映射表维护最新的已提交的逻辑地址到物理地址的映射$\langle LBA, PBA\rangle$.新映射表维护当前正在运行的事务内的更新操作产生的逻辑地址到物理地址的映射$\langle LBA, PBA\rangle$.事务表存储每个事务的当前状态(即运行、提交或回滚).由于采用哈希索引直接映射表,所以直接映射表中可以不实际

存储逻辑地址 LBA. 由于事务提交或者回滚之后，事务的最终状态写入到相变存储器中，所以实际上内存中只需要记录正在运行的事务 ID 即可.

如图 3 所示，外存存储器主要由两个部分组成，数据存储在闪存中，事务最终状态存储在相变存储器中. 对于闪存，每个闪存数据页包括一个数据区和空闲区，分别用于存储用户数据和一些元数据. 空闲区元数据包括该数据页的逻辑地址 LBA，产生该数据页的事务 ID，指向该数据页前一个版本的指针 $Link$(为空表示该数据页是第 1 个版本)，通过该指针把所有版本的数据页链接起来，可记录数据页的更新历史. 对于相变存储器，我们只记录每个事务的事务 ID 和事务的最终状态(C/A，如果为 1 代表事务最终提交；如果为 0 代表事务最终回滚).

3.2 正常事务处理

基于 SPFP 模型，我们详细描述正常事务处理，包括数据页更新、事务提交和事务回滚.

数据页更新. 当一个事务 T^*(ID 为 xid)更新逻辑地址为 lpa、物理地址为 ppa 的数据页 pp，SPFP 执行以下 3 个步骤：(1) 如果事务 T^* 不存在于事务表中，把事务 T^* 的 ID 及其状态〈xid, In progress〉插入到事务表中；(2) 在闪存上创建一个影子页 pp'，物理地址为 ppa'，同时把逻辑地址 lpa、产生该影子页的事务 T^* 的唯一标识 xid 以及该影子页前一个版本的物理地址 ppa 插入到该影子页空闲区的 LBA、XID 和 $Link$ 区域；(3) 把〈lpa，xid，ppa'〉插入到新映射表中，用以在事务结束之后保证直接映射表的正确性.

事务提交. 当一个事务 T^* (ID 为 xid)提交，SPFP 执行以下 3 个步骤：(1) 把〈xid，1〉写入到相变存储器中；(2) 把新映射表中所有属于事务 T^* 的〈lpa，xid，ppa〉合并到直接映射表中，替换掉直接映射表中具有相同逻辑地址 lpa 的〈lpa，ppa〉，同时把这些〈lpa，xid，ppa〉从新映射表中删除；(3) 从事务表中删除事务 T^*.

事务回滚. 当一个事务 T^*(ID 为 xid)回滚，SPFP 执行以下 3 个步骤：(1) 把〈xid，0〉写入到相变存储器中；(2) 把新映射表中所有属于事务 T^* 的〈lpa，xid，ppa〉从新映射表中删除；(3) 从事务表中删除事务 T^*.

3.3 闪存空间回收

当闪存设备的空闲空间低于预设的阈值，系统就会触发垃圾回收模块，回收废旧页. 废旧页包括以下 3 种类型：未提交事务产生的数据页、回滚事务产生的数据页、提交事务产生的已过时的数据页. 事务状态在识别前两种数据页的过程中起到了至关重要的作用. 事务状态可以通过读取相变存储器快速获得，因为相变存储器的读速度接近于内存的读速度. 对于第 3 种数据页，必须要结合相变存储器上存储的事务状态和闪存数据页空闲区的事务唯一标识 XID 以及指向前一个版本的指针 $Link$ 等信息才能完成识别. 垃圾回收如算法 1 所示.

算法 1. 垃圾回收.

```
FOR 每个闪存数据页 DO
  lpa,xid,link=该页空闲区 LBA、XID、Link 域值;
  ppa=该页物理地址;
  stat=读取相变存储器获取事务 xid 的状态;
  IF stat==0 THEN
    释放 ppa 对应闪存页;
    把 ppa 加入到空闲页列表中;
  ELSE
    释放 link 对应闪存页;
    把 link 加入到空闲页列表.
```

对于每个闪存数据页，首先要获取该页空闲区的元数据 LBA、XID、$Link$ 的值，然后再读取相变存储器获取创建该页的事务状态. 如果事务状态为 0，即事务未提交或已回滚，则当前数据页无效，则可以对该数据页进行回收，并把该页地址加入到空闲页列表中以备后用. 如果事务状态为 1，即事务已经被提交，因为可以确保该数据页的前一个版本无效，所以可以回收 $link$ 指向的闪存数据页，并把 $link$ 加入到空闲页列表中以备后用.

3.4 数据库故障恢复

遇到系统正常关机或系统故障后重启数据库时，恢复过程被触发. 它意在恢复最新版本的数据页以及重建直接映射表. 重建直接映射表的目的意在为用户索引到正确的数据.

在恢复过程中，数据页的有效/无效识别是关键，根据 3.3 节描述可知无效的数据页可以通过事务状态和数据页空闲区元信息识别出来. 识别出无效数据页之后剩下的就是有效数据页. 然后，读取有效数据页获取地址映射信息构建直接映射表. 在恢复过程中引入一个数据结构，即物理页状态位图，如图 4 所示. 在物理页状态位图中，每一位对应一个物理页，用以标识这个物理页是有效还是无效. 如果为 1，对应物理页有效；如果为 0，对应物理页无效. 默认情况下，所有闪存数据页都是有效地，即物理页状态位图为全 1. 物理页状态位图只在数据库系统故障恢复时才创建使用，系统正常运行时不需要创建

维护. 恢复过程如算法 2 所示.

Status	1	0	0	0
PBA	(0,0)	(0,1)	(0,2)	(0,3)

图 4　物理页状态位图(*PBA-bitmap*)

算法 2.　恢复.

初始化 *PBA-bitmap* 为全 1;
FOR 每个闪存数据页 DO
　lpa,*xid*,*link*=该页空闲区 *LBA*、*XID*、*Link* 域值;
　ppa=该页物理地址;
　stat=读取相变存储器获取事务 *xid* 的状态;
　IF *stat*==0 THEN
　　根据 *ppa* 设置 *PBA-bitmap* 的相应位为 0;
　ELSE
　　根据 *link* 设置 *PBA-bitmap* 的相应位为 0;
FOR *PBA-bitmap* 中的每个位 DO
　IF 这一位为 1 THEN
　　ppa=该位对应的物理地址;
　　lpa=获取 *ppa* 页空闲区 *LBA* 域值;
　　把〈*lpa*,*ppa*〉插入到直接映射表.

在恢复过程之初,首先要初始化 *PBA-bitmap* 为全 1,即所有闪存数据页都有效. 在第 1 个循环中,通过读取每个闪存数据页的空闲区元数据获取创建该页的事务 *xid*、该页对应的逻辑地址 *lpa* 和该页的前一个版本地址 *link*,然后根据事务 *xid* 读取相变存储器获取该事务的最终状态. 如果事务状态为 0,即事务未提交或已回滚,则当前数据页无效,则置 *PBA-bitmap* 的第 *ppa* 位为 0;如果事务状态为 1,即事务已经被提交,因为可以确保该数据页的前一个版本无效,则置 *PBA-bitmap* 的第 *link* 位为 0. 最终获得每个闪存数据页的最终状态位图 *PBA-bitmap*. 最后,根据 *PBA-bitmap* 的每一位信息来重建直接映射表. 如果位信息为 1,则该位对应的 *ppa* 闪存数据页有效,读取该页空闲区的 *LBA* 信息 *lpa*,把〈*lpa*,*ppa*〉插入到直接映射表中. 虽然某些闪存数据页需要进行两次读操作,尤其是第 2 次读还是随机读操作,但是闪存的随机读顺序读性能都比较好,所以对于性能的影响并不是很大.

4　扩展数据库恢复模型 SPFLP

非强制缓冲区管理策略是数据库管理系统中的一种常用策略. 非强制缓冲区管理允许数据页在事务提交之后刷写到外存上,进而大大提高缓冲区性能. 非强制缓冲区管理策略需要在外存记录事务操作的详细信息. 日志是记录事务操作详细信息的一种方式,但是日志多通过顺序写和顺序读的方式完成日志记录和事务恢复. 而相变存储器支持像内存一样的随机操作,并且相变存储器具有较好的读性能,而写性能也要比闪存要好很多,所以本部分提出了一种基于 SPFP 的优化的能够支持非强制缓冲区管理的数据库事务恢复模型 SPFLP.

4.1　SPFLP 模型架构

如图 5 所示,SPFLP 模型与 SPFP 模型基本上相同,唯一的不同点在于相变存储器上的数据结构需要重新设计以达到记录事务更新操作的详细信息的目的. 在 SPFP 中,相变存储器只需要存储事务 ID 和事务最终状态标识位. 在 SPFLP 中,相变存储器不但要记录事务 ID 和事务最终状态,还需要记录事务内每个更新操作的操作类型(插入/更新/删除)和更新操作内容. 事务内每个更新操作作为链表的一个节点,所有更新操作构成一个链表,链接到事务 ID 和事务最终状态信息后面. 事务开始时,在相变存储器上加入一个不带事务状态的链表头. 对于 SPFP 模型和 SPFLP 模型,相变存储器上的事务状态信息和闪存数据页空闲区的元信息的产生及操作过程是一样的,闪存空间回收算法仍可使用算法 1,这里就不再赘述.

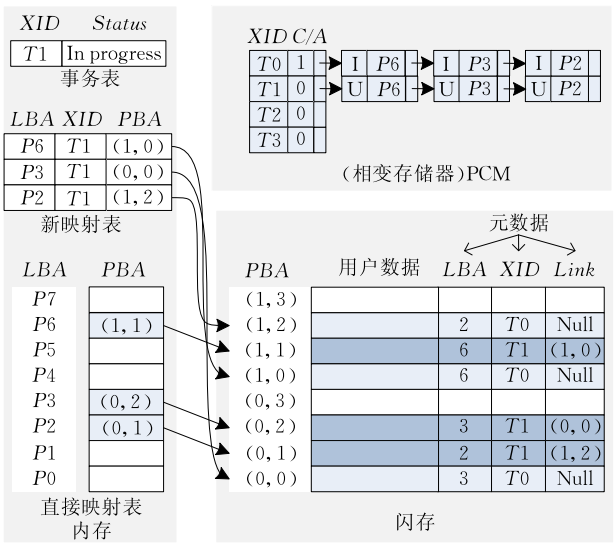

图 5　支持非强制缓冲区管理的数据库恢复模型 SPFLP

4.2　正常事务处理

基于 SPFLP 模型,我们详细描述正常事务处理中的数据页更新和事务回滚. 事务提交过程与 SPFP 模型中的事务提交完全相同.

数据页更新. 当一个事务 T^*(ID 为 *xid*)中更新操作(操作类型为 *U*)对逻辑地址为 *lpa*、物理地址为 *ppa* 的数据页 *pp* 进行操作时,SPFLP 执行以

下 4 个步骤：(1) 如果事务 T^* 不存在于事务表中，把事务 T^* 的 ID 及其状态〈xid，In progress〉插入到事务表中；(2) 在闪存上创建一个影子页 pp'，物理地址为 ppa'，同时把逻辑地址 lpa、产生该影子页的事务 T^* 的唯一标识 xid 以及该影子页前一个版本的物理地址 ppa 插入到该影子页空闲区的 LBA、XID 和 $Link$ 区域；(3) 把操作类型和操作内容形成一个链表节点，然后插入到相变存储器的事务 xid 之后的链表中；(4) 把〈lpa，xid，ppa'〉插入到新映射表中，用于事务结束之后保证直接映射表的正确性.

事务回滚. 当一个事务 T^*(ID 为 xid) 回滚，SPFLP 执行以下 4 个步骤：(1) 把〈xid，0〉写入到相变存储器中；(2) 把相变存储器上的 xid 对应的链表空间释放；(3) 把新映射表中所有属于事务 T^* 的〈lpa，xid，ppa〉从新映射表中删除；(4) 从事务表中删除事务 T^*.

4.3 数据库故障恢复

在 SPFLP 中仍然需要借用 *PBA-bitmap* 这个数据结构来完成数据库故障恢复，即重建直接映射表. 非强制缓冲区管理策略可能导致已经提交的数据页在数据故障前未来得及刷出到闪存上，进而导致数据丢失，所以只能通过相变存储器上的事务更新操作记录来重建数据，进而获取正确的直接映射表. 恢复过程如算法 3 所示.

同算法 2，要设置 *PBA-bitmap* 为全 1，然后通过读取相变存储器和闪存数据页空闲区元数据可以判断数据页的有效/无效，然后根据数据页的有效无效进行其它操作. 对于未提交或已回滚事务产生的数据页，则置 *PBA-bitmap* 的第 ppa 位为 0；对于已提交数据页，同样置 *PBA-bitmap* 的第 $link$ 位为 0. 由于 ppa 闪存数据页已经存在于闪存之上，所以相变存储器上的产生 ppa 的操作记录已经无效，所以需要把产生 ppa 闪存数据页的事务操作记录从相变存储器中的事务 xid 后的链表中删除并释放空间.

算法 3. 恢复(SPFLP).

```
初始化 PBA-bitmap 为全 1;
FOR 每个闪存数据页 DO
  lpa,xid,link=该页空闲区 LBA、XID、Link 域值;
  ppa=该页物理地址;
  stat=读取相变存储器获取事务 xid 的状态;
  IF stat==0 THEN
    根据 ppa 设置 PBA-bitmap 的相应位为 0;
  ELSE
    根据 link 设置 PBA-bitmap 的相应位为 0;
    把产生 ppa 闪存数据页的事务操作记录从相变存储器中的事务 xid 后的链表中删除并释放空间;
FOR 相变存储器中的每个事务 DO
  IF 事务状态为 1 THEN
    FOR 每个链表节点 DO
      重新执行更新操作内容产生相应的闪存数据页;
      修改前闪存数据页对应的 PBA-bitmap 位置 0;
      新产生的闪存数据页对应的 PBA-bitmap 位置 1;
      把该链表节点从该链表中删除并释放空间;
  IF 事务状态为 0 Then
    FOR 每个链表节点 DO
    把该链表节点从该链表中删除并释放空间;
FOR PBA-bitmap 中的每个位 DO
  IF 这一位为 1 Then
    ppa=该位对应的物理地址;
    lpa=获取 ppa 页空闲区 LBA 域值;
    把〈lpa,ppa〉插入到直接映射表.
```

如果事务状态为 0，则可以完全释放该事务对应的链表节点. 如果事务状态为 1，则需要根据事务对应的链表中的每个节点的更新操作内容产生相应的闪存数据页，把修改前的闪存数据页对应的 *PBA-bitmap* 位置 0，并且把新产生的闪存数据页对应的 *PBA-bitmap* 位置 1，最后把该链表节点从该链表中删除并释放空间. 最终获得正确完整的闪存数据页的最终状态位图 *PBA-bitmap*. 最后一步和 SPFP 模型中的相同. 虽然在算法 3 中需要对相变存储器进行读写操作，但是相变存储器具有接近内存的读性能，并且写性能也介于内存和闪存之间，因此系统整体性能不会受到太大影响.

5 实验结果与分析

本部分通过 TPC-C[17] 基准测试验证本文提出的 SPFP 和 SPFLP 模型的性能. 首先描述实验环境，然后对比全闪存的影子页恢复方法和本文两个模型的性能.

5.1 实验环境

为了更好地展示性能，本文实现了 Trace 驱动的闪存模拟器和相变存储器模拟器，通过参数配置可以模拟 MLC NAND 闪存芯片和相变存储器. 实验平台采用 Lenovo 昭阳 K46A，其处理器为 Intel 酷睿 i5 450M，双核 4 线程，内存大小 2 GB. 操作系统使用的是 Fedora14，内核版本为 Linux 2.6.35. 鉴于此，配置相变存储器大小为1 GB，MLC NAND 闪存大小为 32 GB. 保留闪存 10% 的空间用于空间回收，空间回收阈值设置为全部可用空间的 5%. 我们实现了全闪存的影子页恢复技术(SP)、采用组提

交的全闪存的影子页恢复技术(GSP)、SPFP 和 SPFLP 4 种事务恢复模型，事务并发处理采用严格的两阶段加锁机制. 缓冲区采用常用的缓冲区替换策略 LRU. 在 PostgreSQL8.4 上运行 TPC-C 事务并记录数据访问请求，获得标准的在线事务处理负载，即 TPC-C 负载 Trace. 我们利用 50 个客户端和 20 个仓库产生该 Trace，用于测试 SP、GSP、SPFP 和 SPFLP 的性能. 同时，事务的回滚比率设置为默认值 5%. 闪存和相变存储器的参数详见表 1. 其它参数详见表 2.

表 2　实验设置参数

参数	逻辑页大小	缓冲池大小	组提交策略
参数值	4 KB	512	100/组

5.2　性能对比分析

从事务吞吐(单位时间内处理事务数)、事务执行时间(Trace 的总执行时间)、空间回收代价(空间回收执行时间)和恢复代价(恢复执行时间)4 个方面对比了 SP、GSP、SPFP 和 SPFLP 4 种恢复模型. 实验对比数据如图 6～图 9 所示.

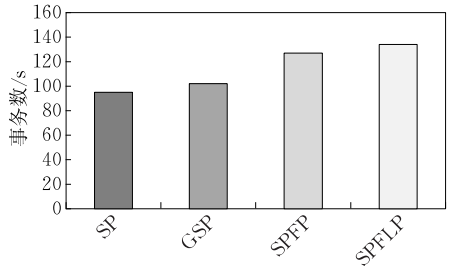

图 6　事务吞吐性

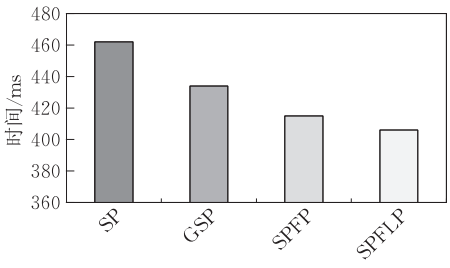

图 7　事务执行时间

如图 6 和图 7 所示，SPFP 的事务吞吐和事务执行时间都要优于 SP 和 GSP，因为相变存储器的读写速度要远高于闪存的读写速度. 采用组提交策略的 GSP 会丢失一定量的已提交数据. SPFLP 的吞吐能力和事务执行时间也要略优于 SPFP. 虽然在 SPFLP 中更新操作会产生大量对相变存储器的写操作，但是 SPFLP 所采用的非强制缓冲区管理策略却极大地减少了对闪存的写操作，并且闪存的写操作延迟要远大于相变存储器的写操作延迟，所以整体上来说 SPFLP 性能还是优于 SPFP 的事务并行处理性能.

由于在获取事务状态过程，SP 要比 GSP 读取更多的闪存页，所以 SP 空间回收代价和数据库恢复代价要大于 GSP 空间回收代价和数据库恢复代价，如图 8 和 9 所示. 在获取事务最终状态时，SPFP 和 SPFLP 都不需要读闪存，所以 SPFP 和 SPFLP 的空间回收代价和数据库恢复代价要低于 SP 和 GSP 的空间回收代价和数据库恢复代价. 由于 SPFP 和 SPFLP 采用同样的空间回收机制，SPFP 和 SPFLP 具有相同的空间回收代价，如图 8 所示. 如图 9 所示，SPFLP 的恢复代价大于 SPFP 的恢复代价，主要原因有以下 3 点：(1) 相变存储器上的事务信息增多，SPFLP 记录了事务更新操作的所有信息；(2) SPFLP 在完成恢复过程中需要删除和释放节点空间；(3) SPFLP 在完成恢复过程中需要改变链表节点中指向下一个节点的指针. 而由表 1 可知，相变存储器的写性能和读性能要相差两个数量级，所以会消耗更长的时间来完成系统故障恢复.

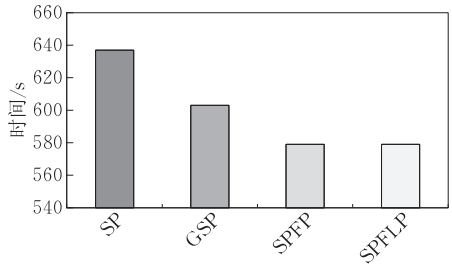

图 8　空间回收代价

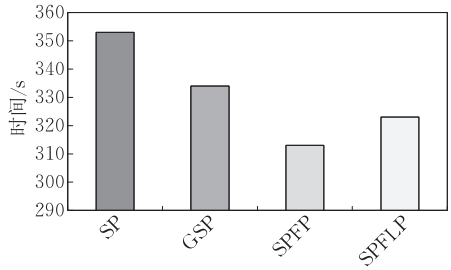

图 9　数据库恢复代价

6　结　论

相变存储器同时具有内存、磁盘和闪存的优良特性，在数据库系统中充分发挥相变存储器的优势特性以提高整个数据库系统的性能成为了新的研究热点. 本文基于相变存储器和闪存的混合存储架构提

出一种基于影子页的数据库事务恢复模型 SPFP,利用相变存储器进行事务管理.随后基于 SPFP 提出了一种支持非强制缓冲区管理策略的数据库事务恢复模型 SPFLP,利用相变存储器跟踪并记录事务更新操作.通过实验证明 SPFP 和 SPFLP 模型能够大大提高事务处理能力.

参 考 文 献

[1] Gray J, Fitzgerald B. Flash disk opportunity for server applications. Queue, 2008, 6(4): 18-23

[2] Condit J, Nightingale E B, Frost C, et al. Better I/O through byte-addressable, persistent memory//Proceedings of the 22nd ACM SIGOPS Symposium on Operating Systems Principles. Big Sky, Montana, USA, 2009: 133-146

[3] Chen Shi-Min, Gibbons P B, Nath S. Rethinking database algorithms for phase change memory//Proceedings of the 5th Biennial Conference on Innovative Data Systems Research (CIDR 2011). Asilomar, California, USA, 2011: 21-31

[4] Mohan C, Haderle D, Lindsay B, et al. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Transactions on Database System, 1992, 17(1): 94-162

[5] Ramakrishnan R, Gehrke J. Database Management Systems. 3rd. McGraw Hill, 2002

[6] Super Talent Technology, SLC vs. MLC: An Analysis of Flash Memory. http://www.supertalent.com/datasheets/SLC vs MLC whitepaper.pdf

[7] Luc Bouganim, Björn Þór Jónsson, Philippe Bonnet. uFLIP: Understanding flash IO patterns//Proceedings of the 4th Biennial Conference on Innovative Data Systems Research (CIDR 2009). Asilomar, CA, USA, January, 2009: 1-12

[8] Chung Tae-Sun, Park Dong-Joo, Park Sangwon, et al. A survey of flash translation layer. Journal of Systems Architecture-Embedded Systems Design(JSA), 2009, 55(5-6): 332-343

[9] Ma Dong-Zhe, Feng Jian-Hua, Li Guo-Liang. LazyFTL: A page-level flash translation layer optimized for NAND flash memory//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2011). Athens, Greece, 2011: 1-12

[10] Micron Corp. 128Mb P8P Parallel Phase Change Memory (PCM) Data Sheet, March 2011

[11] On Sai-Tung, Xu Jian-Liang, Choi B, Hu Hai-Bo, He Bing-Sheng. Flag commit: Supporting efficient transaction recovery in flash-based DBMSs. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(9): 1624-1639

[12] Lu Ze-Ping, Meng Xiao-Feng, Zhou Da. HV-recovery: A high efficient recovery technique for flashed-based database. Chinese Journal of Computers, 2010, 33(12): 2258-2266(in Chinese)
(卢泽萍,孟小峰,周大. HV-Recovery:一种闪存数据库的高效恢复方法. 计算机学报,2010, 33(12): 2258-2266)

[13] Lu Ze-Ping, Qi Xiao-Ying, Cao Wei, Meng Xiao-Feng. LB-Logging: A highly efficient recovery technique for flash-based database//Proceedings of the 13th International Conference on Web-Age Information Management (WAIM 2012). Harbin, China, 2012: 375-386

[14] Lee Sang-Won, Moon Bongki, Park Chanik, et al. Accelerating in-page logging with non-volatile memory. IEEE Data Engineering Bulletin, 2010, 33(4): 41-47

[15] Gao Shen, Xu Jian-Liang, He Bing-Sheng, et al. PCMLogging: Reducing transaction logging overhead with PCM//Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM 2011). Glasgow, United Kingdom, 2011: 2401-2404

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

[17] TPC Benchmark C. Standard Specification. http://www.tpc.org/tpcc/spec/tpcc-current.pdf

FAN Yu-Lei, born in 1984, Ph. D. candidate. His current interests include storage management, query processing and optimization, index and 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, cloud data management, mobile data management, social computing, flash-based systems, privacy-preserving.

Background

Due to the superiority of flash memory, it as a storage alternative for mobile computing devices has been steadily expanded into personal computer and enterprise server markets with the extensive applications of flash memory, it has

attracted more and more attention by the academia and industry. In addition to the excellent characteristics of flash memory, there are many limits in flash memory, such as out-of-place update, page-level read/write, block-level erase, and so on. However, Phase Change Memory (PCM) exhibits better performance than that of flash memory and hard disk, existing database systems may not be able to take full advantage of PCM and flash memory

We consider transaction recovery model for flash or/and PCM-based database systems. Now there are a few of works on this issue, and the works have been published is either using a special log structure, or exploiting the unique characteristic of partial page programming of SLC NAND flash memory, or optimizing transaction recovery model which are not take full advantage of PCM and flash memory simultaneously.

Our research group focuses on the design and implementation for flash-based database systems. We have published several papers about storage management, buffer management, transaction processing, logging, index and query processing for flash-based databases on internal and external conferences. And this work is the first one about transaction recovery model based on PCM and flash memory in our group. This paper is used for improve the transaction recovery performance for flash-based databases.

In this paper, we detail the unique characteristics of flash memory and PCM and consider the storage structure of RAM, PCM and flash memory. Based on hybrid storage of PCM and flash memory, we present a new recovery model, SPFP, which exerts advantages of PCM for managing transaction. For supporting no-force buffer management, we propose an optimized transaction recovery model, SPFLP, which records users' update information in PCM in additions to transactions' information. Our preliminary experiments show that there is a bigger improvement of SPFP and SPFLP in transaction processing performance of databases based on PCM and flash memory in comparison with these based on flash memory

This research was partially supported by the National Natural Science Foundation of China (Nos. 61070055, 91024032, 91124001), the National High Technology Research and Development Program (863 Program) of China (Nos. 2012AA010701, 2013AA013204), the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University (No. 11XNL010).