一种可定制模块化的闪存转换层的设计与实现

杜溢墨，肖侬，刘芳，陈志广
（国防科学技术大学计算机学院，410073，长沙）

摘要：针对闪存转换层(FTL)的具体设计和实现细节不公开问题，提出了一种模块化的FTL设计，将地址映射、垃圾回收、损耗均衡等几个重要部分设计成不同的模块，并提供可定制接口，在每个模块上都可以针对不同的应用定制不同的算法.用文件系统模拟了一个Flash Memory的存储芯片阵列，并将FTL设计部署在该模拟器上，这样就构成了一个完整的SSD模拟器，为研究FTL中的各种算法和机制以及开发基于SSD的各种应用提供了一个实验平台.通过实验测试了FTL设计的性能，并进行了分析.

关键词：闪存转换层；固态盘；地址映射；垃圾回收；损耗均衡
中图分类号：TP333 **文献标志码**：A **文章编号**：0253-987X(2010)08-0042-06

A Customizable and Modular Flash Translation Layer (FTL) Design and Implementation

DU Yimo, XIAO Nong, LIU Fang, CHEN Zhiguang
(School of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: Flash translation layer (FTL) is the key technique of flash-based SSD. There are many productions of SSD, but their detailed design and implementation are kept as a secret. A customizable and modular FTL design is presented in detail which divides the FTL design into several different parts and supplies an interface to make modification for special applications. A file system is employed to simulate a flash memory array, and the FTL design is implemented on the simulator to form a complete simulator of SSD, which provides a platform for users who make researches on algorithms and mechanisms of FTL and on applications based on SSD. The properties of the FTL design are evaluated through experiments and analyzing.

Keywords: flash translation layer; solid state disk; address mapping; garbage reclamation; wear leveling

固态盘(Solid State Disk，简称SSD)提供了类似磁盘的块级接口，存储实体是闪存阵列.在接口与存储实体之间是SSD控制器，用来管理SSD上的RAM和通过Flash控制器访问Flash Memory上的存储单元.与硬盘相比，Flash Memory有以下三个问题：①修改数据不能直接在原数据上重写，导致非定点更新；②数据读写的基本单位是页，而擦除的基本单位是块；③由于寿命问题，每个擦除块只能被擦除有限的次数. 闪存转换层(Flash Translation Layer，简称FTL)的引入就是为了解决这几个问题，它主要包括地址映射、垃圾回收、损耗均衡等部分.当然，这几个部分并不是独立的，例如，损耗均衡可以在地址映射时利用数据布局算法来实现，也可以在垃圾回收时通过合理地选择擦除块来实现.

目前，SSD已经引起了国内外工业界和学术界的广泛关注.在工业界，各种SSD的产品层出不穷，

收稿日期：2010-02-09. 作者简介：杜溢墨(1986—)，男，硕士生；肖侬(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金重点资助项目(60736013)；国家自然科学基金资助项目(60903040)；国家高技术研究发展计划资助项目(2006AA01A106)；新世纪优秀人才支持计划资助项目(NCET-08-0145).

但是在他们提供的产品说明书中，FTL 的设计与实现细节作为 SSD 的核心技术则很难被找到，而且各厂商为了宣传自己的产品难免会有些片面. 在学术界，关于 FTL 的论文也有很多，从地址映射、垃圾回收到损耗均衡都有专门的研究，但却很少有完整的实现和实验测试. 本文提出了一种模块化的 FTL 设计，将地址映射、垃圾回收、损耗均衡等几个重要部分设计成不同的模块，并提供可定制接口，然后通过模拟实验测试了 FTL 的设计并进行了分析.

1　SSD 和 FTL 的相关研究工作

Agrawal 等[1]基于模拟实验比较分析了 SSD 的各种设计策略，并根据它们对性能的影响做了权衡，包括并行度、页大小、条带化程度等. Dirik 和 Jacob[2]在模拟器上进行了大量的实验，指出了 SSD 的性能瓶颈，并测试了从设备级到系统级之间的各个级别增大并行度所带来的性能改善. Chen 等[3]用实验比较了几种 SSD 产品的性能以及其他特点，展示了许多厂商在说明书中没有提到或者与说明书不太符合的一些特征.

Gal 和 Toledo[4]总结了 2005 年以前关于 Flash Memory 的各种典型算法和数据结构，其中也涉及到了 FTL 设计，包括地址映射、垃圾回收和损耗均衡，但是这篇论文属于报告性质，只是分门别类的介绍，并没有给出完整的 FTL 设计.

地址映射作为 FTL 的核心部分，是垃圾回收和损耗均衡的基础，包含许多重要的数据结构和算法，例如 Gupta 等[5]提出的 DFTL 是在页级映射上做的改进，其他地址映射方法则主要针对页级和块级的混合映射结构，包括 BAST、FAST、LAST 等[6-8].

Birrell 等[9]针对高性能 SSD 提出一种基于页映射的 FTL 设计策略，并给出了部分数据结构，对上电逻辑进行了详细的描述，但是其他部分提到的不多，而且没有实现和测试部分.

2　FTL 的设计与实现

本文提出的 FTL 设计的软件体系结构如图 1 所示. 软件采用模块化设计，主要模块的功能描述如下.

由于本文的 FTL 设计针对的是高性能 SSD，因此地址映射完全采用了页映射方式[9]，在 RAM 中保存有完整的页映射表，通过查找映射表可以直

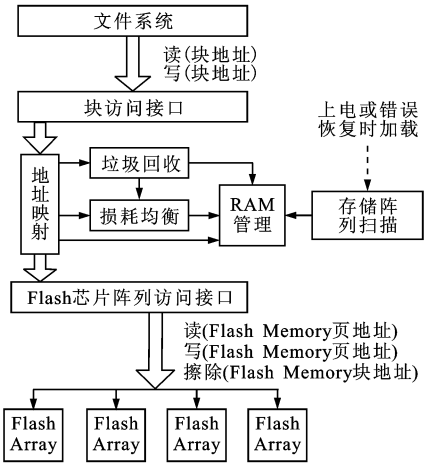


图 1　FTL 设计的软件体系结构

接定位到页，与块级映射和混合映射方法相比，大大缩短了寻址时间. 但是，页映射需要保存所有页的映射关系，对 RAM 空间的消耗很大. 本设计需要的数据结构分为易挥发和不易挥发的，分别存储在 RAM 和 Flash Memory 上. RAM 上的主要数据结构见表 1. Flash Memory 上存储的数据结构记录当前 Flash Memory 各单元的元数据信息，可以在上电时由图 1 所示的存储阵列扫描模块读入 RAM 中建立如表 1 所示的数据结构. Flash Memory 上的元数据有两种类型. 一种是存储在每一页上的 Spare Data 区域，其设计格式见表 2，该区域一般在每一页开头的 64 个字节，其后才是存储实际数据的区域. 在表 2 中，EDC 区域有 20 个字节(16-35)，ECC 区域有 29 个字节(36-64). 在本设计中，页大小(不包括 Spare Data 区域)是 512 字节，EDC 部分只用了 10 位，ECC 部分只用了 14 位，而剩下的都是预留位. 因为页大小是可以变化的，通常是 512 字节的倍数，如 2 倍、4 倍等，当采用更大的页，或者更复杂的校验算法时，EDC 和 ECC 所需要的位数也增多. 另一种是存储在每一块上的一页，在本文中，每个擦除块的最后一页用来作为元数据存储区域，记录所在块上数据的信息摘要，包括块上所有页的逻辑地址与物理地址的映射，这样一旦最后一页形成，通过扫描这一页即可获得块上所有页的映射信息，大大缩短了扫描时间.

2.1　存储阵列扫描模块

存储阵列扫描模块是在上电后最先运行的模块，也是错误恢复时需要启动的模块，见图 2. 它实现了 Scan 函数，在上电后扫描整个 Flash Memory 存储阵列，在 RAM 中构建所需要的所有数据结构.

表 1 RAM 上的主要数据结构

名称	功能描述	类型
LBATable	地址映射表,保存逻辑地址到物理地址的映射	数组,元素个数为存储阵列的总页数
FreeBlocks	空闲块标记表,标记每块是否为空闲	数组,元素个数为存储阵列的总块数
BlockUsage	有效页个数表,记录每块上有效页的个数	数组,元素个数为存储阵列的总块数
ActivePage	当前可以写入的页	结构体,标识 Flash Memory 中的一个地址

表 2 Spare data 区域的格式设计

字节号(1-64)	功能描述	备注
1	坏块标识	每块的第一页中
2	封装标识	每块的第一页中
3-6	块序号	每块的第一页中
7-10	块的擦除次数	每块的第一页中
11	该页是否有效的标识	每块的所有页上
12-15	该页对应的逻辑地址	每块的所有页上
16-35	EDC(Error Detect Code)	每块的所有页上
36-64	ECC(Error Correct Code)	每块的所有页上

2.2 块访问接口模块

FTL 层的块访问接口模块为上层文件系统提供读和写 2 个接口. 读操作与以硬盘为存储介质的操作区别不大,主要还是定位和读取. 但是,由于 Flash Memory 的非定点更新以及写之前需要擦除整个块,导致写操作与以硬盘为存储介质的写有很大不同. 下面对写接口的逻辑进行描述.

步骤 1 当 FTL 层接到写命令后,直接将所要写入的数据和生成的元数据信息写入 ActivePage 中,如果 ActivePage 是块中的第一页,则还需要根据表 2 生成那些只在第一页中包含的元数据信息.

步骤 2 根据写命令中需要访问的逻辑地址查找 LBATable,记录当前 LBATable 中该逻辑地址上对应的物理页地址. 如果记录的该页地址为 0,转到步骤 3;否则,将该页置为无效的元数据信息写入该页,并将该页所对应块的 BlockUsage 个数减一.

步骤 3 在步骤 2 中查找到的 LBATable 位置上记录新写入页的地址,并将新写入页对应块的 BlockUsage 个数加一.

```
//Scan函数
1. 初始化RAM中的数据结构
2. while(还没有扫描到最后一块)
3.         if(最后一页元数据信息完整)
4.                 根据该页上的摘要信息修改RAM上的LBATable
5.         else
6.                 if(第一页信息不完整)
7.                         if(第一页封装完成)
8.                                 将该块加入FreeBlokcs中
9.                         else
10.                                if(第一页为已擦除状态)
11.                                   依次扫描后继页
12.                                    if(后继页均为已擦除状态)
13.                                     封装该块,并将该块加入
                                        FreeBlocks中
14.                                    else
15.                                     该块状态不明确,作废该块
16.                                    endif
17.                                endif
18.                         endif
19                 else//第一页信息完整
20.                        根据第一页上的元数据信息修改LBATable
21.                        依次扫描后继页直到某一页元数据信息
                           不完整//元数据信息完整的页需要根据元
                           数据信息修改LBATable
22.                            if(该页为已擦除状态)
23.                                    将ActivePage指向该页
24.                            else
25.                                    该块状态不明确,作废该块
26.                            endif
27.                endif
28.         endif
29. endwhile
```

图 2 存储阵列扫描模块逻辑伪代码

步骤 4 修改 ActivePage. 分两种情况:如果 ActivePage 的页号等于每块的总页数减一,则根据前面的页生成整个块的摘要信息,写到最后一页上,并查找 FreeBlocks,从中取出一个空闲块(这里可以考虑加入损耗均衡策略),将该块的第一页置为 ActivePage;否则,将 ActivePage 的页号加一即得到新的 ActivePage.

2.3 RAM 管理模块

该模块包括一系列操作 RAM 中数据结构的函数,无论是上电扫描还是进行读写操作时,都需要调用该模块中的各种函数来修改 RAM 中的数据结构. 对包含的主要函数及其功能进行说明,见表 3.

2.4 垃圾回收模块与损耗均衡模块

所谓垃圾回收就是选择需要进行回收的块,将块上的有效数据迁移,然后将整个块擦除,放到空闲块集合中供后续写入的数据使用. 垃圾回收可以被创建成一个新的进程周期地运行,也可以由主程序根据存储状况来调用. 垃圾回收的重点在于如何选择需要回收的块,通常需要考虑两个指标,一是回收效率,二是损耗均衡,根据这 2 个指标的侧重点就可以设计不同的回收策略. 回收效率通常由块上无效

表 3　RAM 管理模块中的主要函数及其功能描述

函数名称	描述
struct path getPath (int blockposition, int pageposition)；	根据 blockposition 和 pageposition 这个二维地址生成所需要可以访问的包含 array 号，chip 号，plane 号，block 号和 page 号的五维的可以访问的地址.
void AddFreeBlocks (int blockposition)	将块 blockposition 对应的 FreeBlocks 置为空闲
void ModifyLBATable (int blockposition, int pageposition)	根据页上的元数据信息修改 LBATable
void AbandonBlock (int blockposition)	作废块 blockposition，即将坏块标识写入该块的第一页上
void RecordActivePage (int blockposition, int pageposition)	将 ActivePage 指向参数中所给的页
void InitiateBlockUsage()	初始化 BlockUsage 数组
void Invalid (Struct path OriginalPath)	将 OriginalPath 指向的页置为无效

页的个数占总页数的比例来衡量，比例越高，擦除时需要迁移的页越少，效率越高. 损耗均衡一般由所选块的擦除次数和损耗最严重块的擦除次数之差来衡量，差值越大，说明该块越应该被擦除来使总体损耗趋于均衡. 在选择需要擦除的块时，为了综合两方面的考虑，用式(1)(其中 i、j 均表示块号，Score(j)表示第 j 块的得分，ρ 为调整因子，obsolete(j)表示第 j 块上无效页的个数，total(j)表示第 j 块上的总页数，erasure(j)表示第 j 块的擦除次数)给每一个候选擦除块打分，按照得分由高到低排序作为擦除的顺序. 通过调整公式中的参数 ρ，可以动态调整效率与损耗均衡的侧重点.

$$\mathrm{Score}(j)=\rho\frac{\mathrm{obsolete}(j)}{\mathrm{total}(j)}+(1-\rho)(\max_{i\geqslant 1}\{\mathrm{erasure}(i)\}-\mathrm{erasure}(j)) \quad (1)$$

采用主程序根据存储状况调用垃圾回收模块的方法，当空闲块数小于总块数的 5%时调用该模块，每次垃圾回收时将所有的垃圾块(块上没有空闲页且存在无效页)都回收，其逻辑描述如图 3 所示.

除了在垃圾回收中考虑损耗因素以达到损耗均衡的目的外，还可以在数据写入时考虑损耗均衡，将

```
//垃圾回收函数
1. while(还有垃圾块)
2. 按照公式(1)给备选擦除打分
3. 选择分数最高的块
4. 将所选块上的所有有效页拷贝到其他空闲页上
5. 擦除该块
6. 将该块所有对应的FreeBlocks值为空闲
7. endwhile
```

图 3　垃圾回收模块逻辑伪代码

冷数据(即不经常访问的数据)分配到损耗较严重的块上，热数据(即经常访问的数据)分配到损耗较小的块上，这样就可以避免损耗严重块的频繁擦写和损耗少的块的长期闲置. 还可以根据数据的访问热度将冷数据动态迁移到损耗严重的块上，这样也可以平衡损耗，但是这种动态迁移的方法需要较大的开销，只有在损耗极不均衡或是系统空闲的时候才采用.

本文的设计全部采用模块化的思想，提供可定制的接口，针对不同的应用可以在本设计的基础上动态添加并选择需要的算法，以实现针对不同应用的 FTL 设计.

3　实验与性能分析

3.1　模拟器

为了测试 FTL 设计的性能. 我们用文件系统模拟了一个 Flash Memory 存储阵列，并给上层提供了读写页和擦除块的接口. 用文件的树状目录结构来模拟 Flash Memory 存储阵列的层次结构，如图 4 所示. 通过修改 array、chip、plane、block、page 的数目参数，可以得到不同大小和层次结构的存储阵列. 该模拟器不仅可以获得读写页和擦除块的延迟，还可以通过访存文件来读写存储在芯片上的数据，就像是有一个真正的 Flash Memory 存储阵列一样. 由于对文件的读写比对实际 Flash Memory 存储芯片的读写慢得多，所以需要通过模拟计算来获得读写延迟，而不是直接使用访存文件的延迟. 当读写操作量很大时，模拟实验会比较耗时. 设定好参数，初始化模拟器，就得到一个类似 Flash Memory 存储阵列结构的文件系统，然后将本文的 FTL 的设计部署在该文件系统之上，就可以通过实验来测试 FTL 设计的性能.

3.2　实验环境

本文的实验环境为目前主流的桌面机，其配置为 4 个 Intel Core™2Quad Q8 200 2.33 GHz 处理器，4GB内存，320GB硬盘，Ubuntu8操作系统，

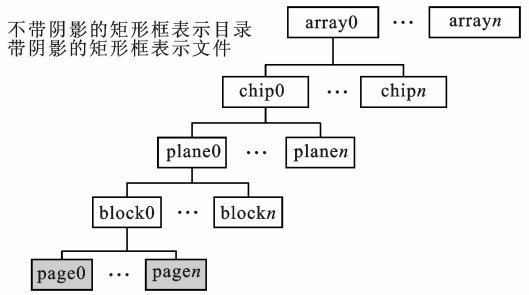


图 4 Flash Memory 存储阵列的文件系统模拟图

内核版本号为 2.6.27,文件系统为 Ext3.

3.3 实验结果

通过调整参数测试不同规模 SSD 的上电扫描时间,结果如图 5 所示. 对每种规模下的 SSD 分别测试出厂扫描时间、经过出厂扫描后的空块扫描时间、SSD 中写入部分数据后的扫描时间以及 SSD 中写满数据后的扫描时间. 如图 5 所示,不同规模下的 SSD 出厂扫描时间差别很大,而后扫描时间相差并不太大. 这是因为出厂扫描时需要将所有的页都扫描一遍,而后的扫描通过每块首页的元数据信息或者每块最后一页的元数据信息就可以得到整个 Flash Memory 存储阵列的状态,并在 RAM 中构建起所需的数据结构,所以扫描时间要比出厂首次扫描短很多.

用 trace[10] 测试 FTL 的读写响应时间,结果见图 6. 横坐标用 Flash Memory 存储阵列的总页数来表示 SSD 的规模,纵坐标表示每个 I/O 的响应时间. 从图中可以看出,无论在哪种规模下,读延迟都大大小于写延迟,而不同规模下的延迟差别不大. 这与 SSD 非对称的读写性能很吻合.

在 trace 运行完后,通过查看每块上记录的擦除次数,列出了各块的损耗情况,见图 7. 从图中我们可以看出,在经常被使用的块之间的损耗基本能

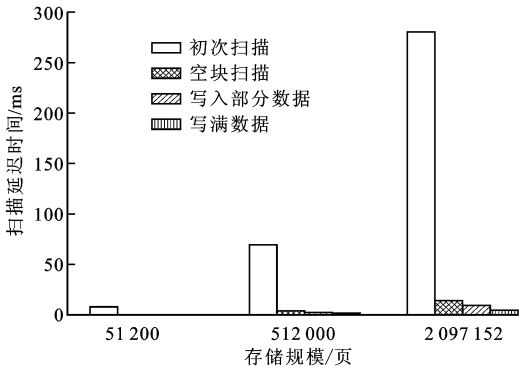


图 5 不同规模下的 SSD 扫描时间

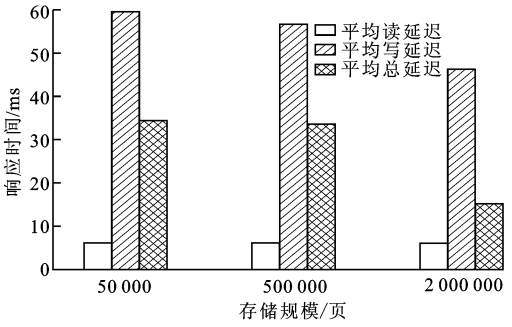
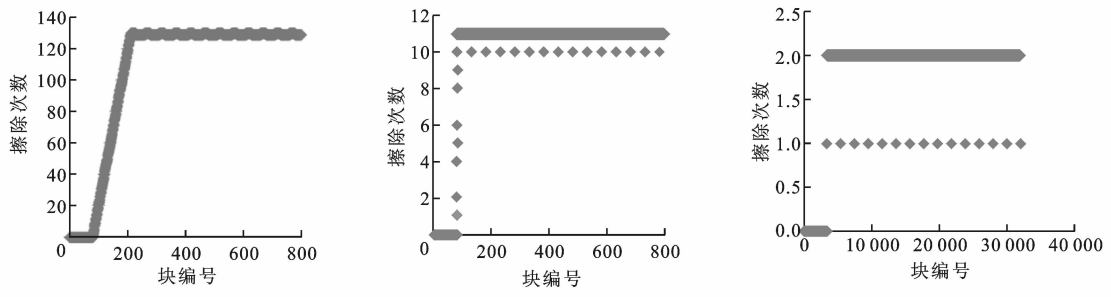


图 6 不同规模 SSD 的平均响应时间

够达到均衡,但是整体损耗并不太均衡,有一些块几乎没有被使用到. 可以按照式(1)调整参数 ρ,选择不同的垃圾回收策略来使这种情况得到改善.

4 结 论

本文给出了一种可定制的模块化的 FTL 设计,并将该 FTL 设计部署在文件系统模拟的 Flash Memory 存储阵列上. 为了方便用户扩展定制自己所需要的算法,各个模块之间不仅低耦合,而且这些模块(包括地址映射、垃圾回收、损耗均衡等)都采用了简单、高效的算法. 通过可定制接口可以添加所需



(a)51 200 页 (b)512 000 页 (c)2 097 152 页

图 7 不同存储阵列规模下的损耗分布图

的算法，通过配置参数可以调整 Flash Memory 阵列的存储规模和组织结构. 最后，本文验证了 FTL 的设计，并用 trace 测试了各种 Flash Memory 存储阵列规模下 FTL 设计的性能.

下阶段的工作有两个方向：一是把本文中 FTL 的设计实现到硬件上，做成一个 SSD 产品，再测试它的性能；二是研究基于 Flash Memory 的 I/O 调度，因为以往的 I/O 调度策略都是针对磁盘的，并没有利用 Flash Memory 随机读性能好的优点，我们可以用本文的 SSD 模拟器进行实验，设计一个专门用于 Flash Memory 的 I/O 调度算法，以提高系统的整体性能.

参考文献：

[1] AGRAWAL N, PRABHAKARAN V, WOBBER T, et al. Design tradeoffs for SSD performance [C]// Proceedings of the 2008 USENIX Technical Conference. Berkeley, USA: USENIX, 2008:57-70.

[2] DIRIK C, JACOB B. The performance of PC solid-state disks (SSDs) as a function of bandwidth, concurrency, device architecture, and system organization [C]// Proceedings of ISCA'09. New York, USA: ACM, 2009:279-289.

[3] CHEN Feng, DAVID A K, ZHANG Xiaodong. Understanding intrinsic characteristics and system implications of flash memory based solid state drives[C]// Proceedings of 2009 ACM SIGMETRICS conference on Measurement and Modeling of computer systems. New York, USA: ACM, 2009:181-192.

[4] GAL E, TOLEDO S. Algorithms and data structures for flash memories [J]. ACM Computing Survey, 2005, 37(2):138-163.

[5] GUPTA A, KIM Y, URGAONKAR B. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings[C]// Proceedings of ASPLOS'09. New York, USA: ACM, 2009:229-240.

[6] KIM J, KIM J M, NOH S H, et al. A space-efficient flash translation layer for CompactFlash systems[J]. IEEE Transactions on Consumer Electronics, 2002, 48(2):366-375.

[7] LEE S W, PARK D J, CHUNG T S, et al. A log buffer-based flash translation layer using fully-associative sector translation. [J/OL]// Transactions on Embedded Computing Systems, 2007, 6(3). [2010-01-09]. http://portal.acm.org/citation.cfm?id=1275986.1275990.

[8] LEE S, SHIN D, KIM Y J, et al. Last: locality-aware sector translation for NAND flash memory-based storage systems [J]. SIGOPS Oper Syst Rev, 2008, 42(6):36-42.

[9] BIRRELL A, ISARD M, THACKER C, et al. A design for high-performance flash disks [J]. SIGOPS Oper Syst Rev, 2007, 41(2):88-93.

[10] CHANG Lipin, KUO Teiwei. An adaptive stripping architecture for flash memory storage systems of embedded systems [C]// Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium. Piscataway, NJ, USA: IEEE, 2002:187-196.

[本刊相关文献链接]

应用变量因果序分析的符号有向图建模方法. 西安交通大学学报，2010，44(5)：85-90.

一种面向写穿透 Cache 的写合并设计及验证. 西安交通大学学报，2010，44(4)：1-4.

利用互易定理的混响声场数值模拟. 西安交通大学学报，2010，44(3)：110-114.

一种分段测量保证 QoS 约束的任播通信模型. 西安交通大学学报，2010，44(2)：44-49.

用属性单值表示的决策表简化算法及属性核计算. 西安交通大学学报，2010，44(1)：87-90.

一种支持负载均衡的存储调度算法. 西安交通大学学报，2009，43(10)：61-65.

CorsairFS：一种面向校园网的分布式文件系统. 西安交通大学学报，2009，43(8)：43-47.

面向存储资源管理的多协议存储系统. 西安交通大学学报，2009，43(6)：10-14.

利用角度测量估计机动目标运动参数的方法. 西安交通大学学报，2009，43(6)：67-71.

面向 Cell 宽带引擎架构的异构多核访存技术. 西安交通大学学报，2009，43(2)：1-5.

一种基于网关的网格数据互操作框架. 西安交通大学学报，2009，43(2)：20-24.

利用最大似然准则的双向联想网络研究. 西安交通大学学报，2008，42(8)：963-966.

一种基于角度测量估计飞行目标运动参数的新方法. 西安交通大学学报，2008，42(8)：986-990.

一种电子设备电磁敏感性的仿真建模与优化设计方法. 西安交通大学学报，2008，42(6)：688-692.

一种 Log-Gabor 滤波结合特征融合的虹膜识别方法. 西安交通大学学报，2007，41(8)：889-893.

（编辑　武红江）