Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2014-CJ-002

一种优化的闪存地址映射方法

张琦，王林章，张天，邵子立

Postprint Version. Originally Published in: 软件学报，2014,25(2):314-325
[doi: 10.13328/j.cnki.jos.004528]

软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(2):314−325 [doi: 10.13328/j.cnki.jos.004528]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种优化的闪存地址映射方法*

张 琦[1], 王林章[1], 张 天[1], 邵子立[2]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210046)

[2](香港理工大学 计算系,香港 00853)

通讯作者: 王林章, E-mail: lzwang@nju.edu.cn

摘 要: 近年来,NAND 闪存广泛应用于各类嵌入式系统.由于“异地更新”的限制,闪存中需要地址映射方法将来自文件系统的逻辑地址转换为闪存中的物理地址.随着闪存存储空间的日益增长,如何使地址映射表占用较小的内存而又不损失较多性能,成为一个重要的问题.基于需求的页级地址映射方法能够有效地解决这个问题,然而该方法会产生地址转换页操作的额外开销,影响系统性能.从基于需求的地址映射方法出发,从两方面进行优化:首先,为了减少转换页的频繁更新,提出了页级地址映射缓存技术以统一在闪存和内存中的地址映射信息的粒度;其次,设计了基于地址转换页的数据聚集技术.通过该技术,每个数据块在垃圾回收时产生的地址转换页的更新开销被降至最低.实验用一系列基准数据集并与之前代表性的工作进行比较,结果表明,优化的地址映射方法能够大量减少额外地址转换页的开销,并提高闪存存储系统的性能.

关键词: 闪存;地址映射;存储系统;嵌入式系统

中图法分类号: TP316 **文献标识码**: A

中文引用格式: 张琦,王林章,张天,邵子立.一种优化的闪存地址映射方法.软件学报,2014,25(2):314−325. http://www.jos.org.cn/1000-9825/4528.htm

英文引用格式: Zhang Q, Wang LZ, Zhang T, Shao ZL. Optimized address translation method for flash memory. Ruan Jian Xue Bao/Journal of Software, 2014,25(2):314−325 (in Chinese). http://www.jos.org.cn/1000-9825/4528.htm

Optimized Address Translation Method for Flash Memory

ZHANG Qi[1], WANG Lin-Zhang[1], ZHANG Tian[1], SHAO Zi-Li[2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210046, China)

[2](Department of Computing, The Hong Kong Polytechnic University, Hongkong 00853, China)

Corresponding author: WANG Lin-Zhang, E-mail: lzwang@nju.edu.cn

Abstract: NAND flash memory has been widely used in various embedded systems. Due to the “out-of-place” update constraints, a component of address translator in NAND flash management is needed to translate logical address from file system to physical address in NAND flash. With the capacity increase of NAND flash, it becomes vitally important to take small RAM print of the address mapping table while not introducing big performance overhead. Demand-based address mapping is an effective approach to solve this problem by storing the address mapping table in NAND flash (called translation pages) and catching mapping items on-demand in RAM. However, in such address mapping method, there exists extra many translation pages that may incur much performance overhead. This paper solves two most important problems in translation page management. First, to reduce frequent translation page updates overhead, a page-level caching mechanism is proposed to unify the granularity of the cached mapping unit in NAND flash and in translation caching. Second, to reduce the garbage collection overhead from translation pages, a translation page based data-assemblage strategy is designed to group data pages corresponding to the same translation page into one data block, reducing the cost of translation page update during garbage collection to the minimal level. The presented scheme is evaluated using a set of benchmarks and is compared to a representative previous

* 基金项目: 国家自然科学基金(61170066, 61321491, 61003025); 国家高技术研究发展计划(863)(2011AA010103)
收稿时间: 2013-05-07; 定稿时间: 2013-09-29

work. Experimental results show that the new techniques can achieve significant reduction in the extra translation operations and improve the system response time.

Key words: flash memory; address translation; storage system; embedded system

随着闪存技术的发展与成熟,闪存存储器已广泛应用于各类存储系统,如 U 盘、智能手机、平板电脑、数字存储卡、固态硬盘等.与传统磁盘存储器相比,闪存具有读写性能高、非易失性、低功耗、高密度和良好的抗震性等优势.因此,闪存已成为移动嵌入式设备的存储介质.然而,闪存也存在一些局限,如“异地更新”和有限的块擦写次数.为了解决这些不足并使闪存像传统块设备一样工作,闪存存储系统中出现了一种称为闪存转换层(flash translation layer,简称 FTL)的嵌入式软件,用于处理外部 I/O 请求及管理闪存[1].其中,地址映射作为一个重要组件,负责将来自于文件系统的逻辑地址转换为闪存中的物理地址.因此,如何有效地执行地址映射并管理映射信息,成为一个关键问题.

闪存转换层中主要有 3 种地址映射方法[1]:页级映射[2−4]、块级映射[5−7]和混合映射[8−11].页级映射以页为单位执行地址映射,可直接定位数据.然而,页级地址映射表保存了所有逻辑页到物理页的映射信息,内存开销很大.在块级地址映射中,块级映射表仅保存逻辑块到物理块的映射信息,内存开销小;然而,映射信息的粒度较大可能导致定位冲突,读写性能较低.在混合映射方法中,物理块被逻辑地分为数据块和日志块,数据块中使用块级映射,同时,利用页级映射定位在日志块中的更新数据.虽然混合映射方法结合了块级和页级映射两者的优点.但它并没有完全解决两者中存在的性能问题.

与块级和混合映射相比,页级地址映射能够高效地执行地址映射,然而,庞大的页级地址映射表产生的内存开销使其难以应用于资源受限的嵌入式系统.近年来提出了一种基于需求的页级地址映射方法(DFTL)[3],用于解决页级地址映射的内存开销问题.在该方法中,完整的页级映射表以地址转换页的形式保存在闪存中,而频繁使用的映射项缓存在内存中.因此,DFTL 能够有效地减少内存开销并且保持页级地址映射的高性能.然而,由于所有更新的地址映射信息都需要写入闪存,这些对地址转换页的大量操作影响了地址映射的性能.Qin 等人提出了一种基于需求的两级映射缓存方法[4],由于充分利用了空间局部性,改善了缓存命中率及系统性能.但该方法仅考虑缓存中的地址映射,而没有优化在缓存和闪存中的整个地址映射过程.Qi 等人也针对 DFTL 进行了优化[12],主要将缓存进行分类,并用日志页缓存对地址映射信息的更新.该方法虽然考虑了映射表的优化,但没有考虑闪存中的地址转换页的更新开销.同时,也出现了部分写缓存优化技术[13,14],但这些技术主要针对数据缓存而非地址映射缓存,因此无法直接应用于地址映射方法.

本文提出了一种优化的基于需求的页级地址映射方法,为文献[15]工作的扩展.主要从两个方面对闪存地址映射进行优化:首先,针对缓存未命中造成的大量地址转换页操作,通过分析闪存中的数据读写特性,设计页级地址映射缓存,统一缓存和闪存中地址映射信息的粒度;其次,为降低垃圾回收产生的大量地址转换页更新开销,提出了基于转换页的数据聚集方法,将属于同一地址转换页的数据写入到相同的数据块.因此,将垃圾回收时的转换页的更新开销降至最低的常量值,提高了系统性能.实验中用 FlashSim[16]仿真了一个 32GB 的 NAND 闪存存储系统,结果表明:与以往的地址映射方法相比,本文提出的优化地址映射方法在稍微增加空间开销的情况下,能够平均降低 90.93%的额外地址转换页操作,并提高 22.14%的系统性能,降低 26.51%的块擦写次数.

本文第 1 节给出闪存存储系统的基本架构和地址映射方法.第 2 节提出优化的地址映射方法.在该方法中,着重说明页级映射缓存和基于转换页的数据聚集方法.第 3 节通过实验验证本方法的有效性.第 4 节总结全文并提出未来的工作.

1 闪存地址映射

NAND 闪存芯片主要由物理块组成,每个物理块分为一定数量的物理页:块是擦写操作的基本单元,而页是读写操作的基本单元.一个页包含了数据域和备用域,备用域又称为 OOB(out of band).OOB 一般用于存储错误纠正码(ECC)和一些自定义信息.表 1 显示了一个典型的 4G NAND 闪存的规格参数.

Table 1 Samsung 4G MLC NAND flash specification

表 1 三星 4G MLC NAND 闪存规格

参数	值
页大小(KB)	2
OOB 大小(B)	64
块大小(KB)	256(data)+8(OOBs)
每块页数	128
读页时间(μs)	60
读 OOB 时间(μs)	20
写页时间(μs)	800
块擦写时间(μs)	1 500
可用块擦写次数(次)	5 000
ECC 编码(bits/B)	4/512

虽然 NAND 闪存具有诸多优势,但也存在一些局限:

- 首先是“异地更新”,当数据页被写入数据后无法再次更新,直到所属的整个块被擦除.因此,对同一逻辑地址的不同更新数据将被写入不同的物理页,其中,存放最新数据的页称为有效页,而之前旧数据页称为无效页.因此,闪存中需要地址映射方法转换逻辑地址到最新数据所在的物理地址;同时,由于数据页不断消耗,需要垃圾回收机制去拷贝有效页并擦除无效数据块,以重新获得空数据块.
- 其次是有限擦写次数的限制,对于单级单元(SLC)NAND 闪存,每个块的擦写次数大约是 100 000 次,而对多级单元(MLC)NAND 闪存,擦写次数大约只有 10 000 次[18].如果块实际擦写次数大于限制则可能成为坏块,无法再被使用.因此,需要均匀损耗机制以平衡不同块的热度,并选取合适的数据块进行垃圾回收.

为了弥补上述缺陷,NAND 闪存中使用闪存转换层(FTL)管理闪存.闪存转换层位于文件系统和闪存驱动层之间,对用户完全透明,并为上层文件系统提供块设备的操作接口.

地址映射作为闪存转换层的重要组件,负责将来自文件系统的逻辑地址转换为闪存中的物理地址.地址映射方法主要分为 3 类:页级映射、块级映射和混合映射.

在页级地址映射中,映射信息粒度与数据读写单元一致,都为数据页,因此数据请求可以写入闪存的任何数据页中.同时,页级地址映射能够延迟垃圾回收触发的时间,也就是说,直到闪存存储接近饱和才触发垃圾回收,因此能够获得更优的系统性能.然而页级地址映射方法需要维护庞大的页级地址映射表,在资源受限的嵌入式系统中难以实现.块级地址映射的映射粒度为数据块,每个数据请求的逻辑地址分为块号和块内偏移量,每个逻辑块被分配一个或多个物理块,然而由于“异地更新”的限制,已存储数据的数据页无法直接更新,因此,块级地址映射存在更新冲突,可能触发大量的垃圾回收,系统性能较低.在混合映射方法中,物理块被分为数据块(primary block)和日志块(log block),数据块中使用块级地址映射映射首次写入的数据,而日志块利用页级地址映射存储更新数据.虽然混合映射权衡了系统性能和内存开销,但是在垃圾回收机制中需要将数据块和日志块进行合并,可能产生大量的性能开销,因此,混合映射方法也没有很好地解决性能和内存开销的问题.近年来提出了基于需求的页级地址映射(DFTL)[8],一方面利用页级地址映射方法提供高效的地址转换性能,另一方面,通过地址映射缓存机制解决了页级地址映射表产生的大量内存开销问题.

DFTL 中的物理块被逻辑地分为两类:数据块(data block)和地址转换块(translation block).数据块用于存储数据,转换块用于保存页级地址映射表.数据块和转换块共享闪存的物理空间.每个地址转换块中的页称为地址转换页(translation page),每个页包含了一定数量的逻辑页号到物理页号的页级地址映射项.为定位到最新的转换页,内存中维护了一个全局转换页映射表(global translation directory,简称 GTD).同时,在内存中使用地址映射缓存(cached mapping table,简称 CMT),以缓存在转换页中频繁访问的地址映射项.地址映射缓存使用 LRU 算法对地址映射项进行置换.由于内存中只维护 GTD 和 CMT,内存开销显著降低.同时,利用数据请求的时间局部性,保持了高效的地址映射性能.

在基于需求的页级映射方法中,对地址转换页的操作将产生额外的性能开销,开销主要来源于两个方面:

- 首先,从闪存的地址转换页中提取地址映射项到缓存中将产生转换页的读操作,当缓存已满需要进行置换时,置换的更新映射项需要先读取对应的转换页,并在更新后写回闪存.因此,缓存未命中时,最好的情况产生 1 个读操作,最坏情况会产生 2 个读操作和 1 个写操作.以图 1 为例,假设数据请求的逻辑地址是 28,由于缓存未命中,首先访问 GTD 获得最新地址转换页的地址,然后从内存中读取该页(TP1),定位到对应的地址映射项(28,56),最后将其放入地址映射缓存中.当 CMT 置换出地址映射项(112,84)时,由于该映射项在缓存中被更新过,需要首先根据 GTD 获得闪存中的地址转换页(TP32),读取该页后更新对应的地址映射项,最后写回到新的转换页中.可以看到,地址映射缓存会带来很多地址转换页读写操作,这些操作穿插在处理数据请求的过程中,影响系统性能.
- 另一方面,性能开销来源于垃圾回收过程.当数据块被垃圾回收时,所有有效页对应的转换页都需要更新.如图 2(a)所示,在回收块中有 4 个有效页(PPN 从 10 到 13),每个页对应于不同的转换页(分别为 TP97,TP100, TP101,TP103).在拷贝有效页后,所有对应的地址转换页也需要被读取、更新、写回到闪存中(写回后分别为 TP128,TP129,TP130,TP131).值得注意的是,实际更新的地址映射信息只有 4 项,却造成对应 4 个转换页的更新操作.因此,当回收块中存在很多有效数据页时,垃圾回收将产生大量的转换页更新开销.

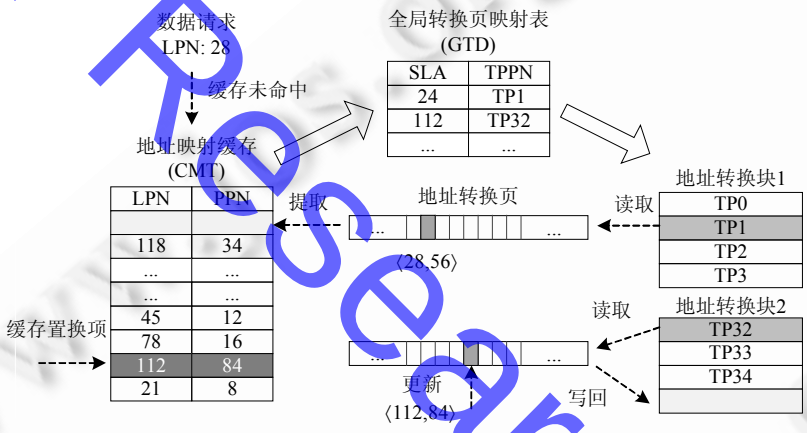
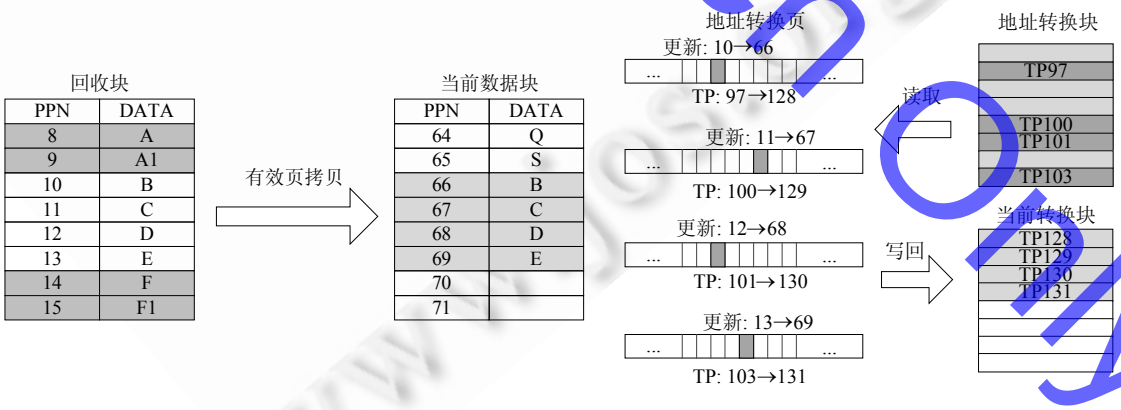


Fig.1 Address mapping cache overhead

图 1 地址映射缓存产生的开销



(a) 数据块回收时的有效页拷贝 (b) 数据块回收时地址转换页的更新

Fig.2 Translation page update during garbage collection

图 2 垃圾回收时产生的转换页操作

总的来说,基于需求的页级地址映射方法存在两个主要问题.首先,地址映射缓存的映射信息利用率较低,在读取整个转换页后,只缓存其中一个地址映射项.对于连续读写,某个地址转换页可能被重复多次请求产生大量开销.其次,在处理写请求时,不同逻辑地址的数据被连续写入同一数据块中,在垃圾回收时将产生大量的转换页更新.针对上述问题,本文设计了一种优化的地址映射方法.

2 基于需求的页级地址映射优化方法

本文从基于需求的页级地址映射方法出发,试图解决地址映射造成的额外性能开销问题.这部分开销主要来源于地址映射缓存以及闪存中的地址转换页.通过分析这两个来源,设计了一种基于转换页的缓存机制、闪存数据写入和垃圾回收的完整优化方法.

2.1 页级地址映射缓存

本文提出了一种页级地址映射缓存技术(page-level address mapping cache,简称 PAMC).缓存的数据单元为整个地址转换页,因此统一了闪存和缓存中的地址映射信息的粒度,并充分利用数据的时间局部性和空间局部性.每个地址转换页包含了 1MB 地址空间的映射信息,如短时间内频繁访问局部数据,只需访问页级地址映射缓存而不需要访问闪存.同时,当某个转换页需要被置换出缓存时,所有更新的地址映射信息能够一并更新到闪存中,提高了映射信息的利用率.

全局地址转换页映射表负责定位闪存中的地址转换页,在读取地址转换页后,可获得实际的地址映射信息.因此,GTD 也是一个页级地址映射表.本文利用内存中的全局转换页映射表和页级地址映射缓存具有相同映射信息粒度这一特征,设计融合两者的访问过程,以提高地址映射性能.本文方法扩展了全局转换页映射表,增加了缓存索引(cache index,简称 CI),从 GTD 中直接定位缓存中的转换页,使转换页映射表直接链接到页级映射缓存.当有逻辑地址请求时,首先根据请求的逻辑地址定位到全局转换页映射表的项,由于每一项的起始逻辑地址是固定的,因此,将逻辑地址除以每个转换页保存的地址映射项数目即能访问到转换页映射表中的项,该项维护了转换页在闪存中的物理地址以及缓存索引.如果缓存索引为–1,则表明该转换页尚未缓存,需要访问闪存获取地址映射信息;否则,根据缓存索引可直接访问缓存的地址映射页以获得请求的物理地址.可以看到,通过本文方法提出的新的地址映射访问方式,简化了地址映射的访问过程,将缓存中的连续或者哈希查找的方式变为直接访问,节约了缓存访问时间.

图 3 为页级地址映射缓存示例,假设每个转换页能够存储 6 个地址映射项,请求读数据的逻辑地址为 36.首先,在全局转换页映射表中定位其映射信息项,通过计算 36/6=6 定位到映射表中的第 6 项,其转换页地址为 94,缓存索引为 11.由于缓存索引不为–1,可直接访问 PAMC 的第 11 项获得实际地址映射(36,134),从而得到对应逻辑地址 36 的物理地址是 134.

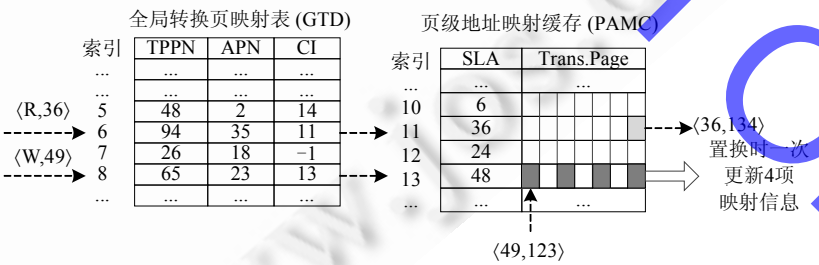


Fig.3 Page-Level address mapping cache

图 3 页级地址映射缓存

本文使用最近最少使用算法(LRU)作为页级地址映射缓存的置换策略,其中,转换页的更新次数也是选择置换的依据之一,如果某个转换页在缓存中从未被更新过,它将被优先置换出缓存,因为这样的置换操作不会产

生额外的转换页读写开销.利用缓存中的置换项对应的地址转换页的起始地址(start logical address,简称 SLA)和全局地址转换页映射表(GTD),以定位闪存中实际存储的转换页地址,执行置换操作.在图 3 的例子中,当有个写请求〈W,49〉时,PAMC 中的第 13 项转换页需要被置换.可以看到,置换时,所有 4 个修改后的映射信息能够一次写入地址转换页.

2.2 基于转换页的数据聚集方法

为了进一步减少因地址映射信息更新带来的转换页开销,本文设计了基于转换页的数据聚集技术.数据写入将以其逻辑地址对应的转换页作为指导,逻辑地址对应于同一转换页的数据将被写入相同的数据块.因此,垃圾回收时,数据块对应的转换页数量最多为常量 1,由此产生的额外地址映射信息更新的开销被降至最低.

为了实现数据聚集技术,本文继续扩展全局转换页映射表,在表中的每一项保存了可用数据页的物理地址(available page number,简称 APN),其中,物理地址为–1 代表该项尚未分配数据块.同时,使用空块池(free block pool,简称 FBP)记录闪存中所有的空数据块,当有数据请求时,按需分配空数据块给全局转换页映射表.在垃圾回收中,擦写后的空块回收到空块池.数据聚集技术优化了数据块中的数据分布,即,每个数据块中数据的逻辑地址限定在对应转换页维护的地址空间范围内,由于地址相近的数据更容易被一起更新,该方法能够间接减少垃圾回收时有效页拷贝数量.

在闪存初始阶段,所有空数据块属于空块池,全局转换页映射表中每项可用数据页地址都为未分配状态(–1),当有写数据请求时,根据上文提到的计算方法定位全局转换页映射表的项,如果项内的 APN 不为–1,则直接将数据写入到对应的数据页,随后更新 APN 为所分配数据块中的下一个可用数据页,如果数据块已满,则将该地址置为–1;否则,若该转换页尚未分配数据块,检查空块池中是否存在可用的空数据块,若存在则按需分配并记录第 1 个数据页的物理地址作为 APN,否则调用垃圾回收获得新的空数据块.

图 4 为数据聚集示例.假设每个块中包含 4 个物理页,每个转换页能保存 6 个地址映射项.对于逻辑地址 36 的写请求,首先计算其全局转换页映射表中的项为 36/6=6,然后得到可用数据页的地址(APN)为 22.因此,数据 D 能够直接写入到数据块 5 的数据页 22 中.写入后,将可用数据页地址更新为 23.对于另一个逻辑地址为 49 的写请求,计算得到第 8 项的 APN 为–1,说明该转换页尚未分配数据块,因此从空块池获得一个空数据块,然后将数据 E 写入空块的第 1 个数据页(数据页 32)中,最后,APN 更新为下一个数据页地址 33.可以看到,对于数据块 5,根据数据聚集方法,对应的地址映射信息全部属于转换页 TP48.这里,TP48 已经在缓存中,因此可直接在缓存中更新地址映射信息.新的空块也只对应转换页 TP65.这样,每个数据块最多只对应 1 个地址转换页,从而将转换页的更新开销降低至最少.

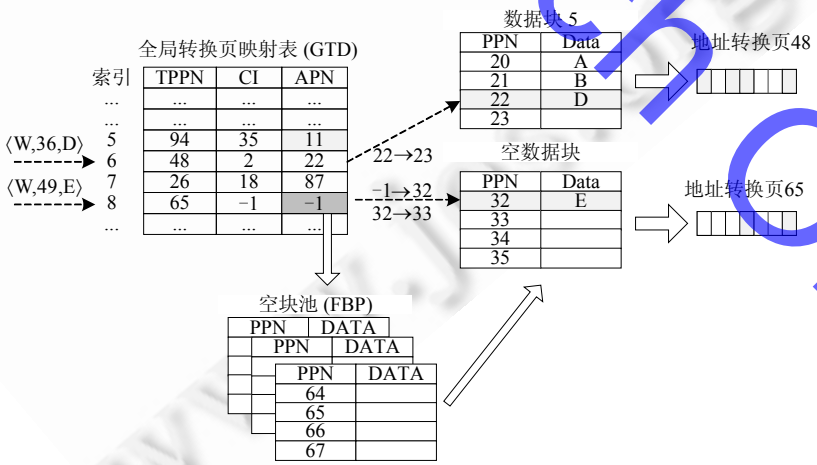


Fig.4 Translation page based data-assemblage

图 4 基于转换页的数据聚集

2.3 基于数据聚集的垃圾回收

由于物理块根据存储数据类型被分为数据块和地址转换块,相对应地存在两种垃圾回收:数据块垃圾回收和转换块垃圾回收.在数据块的垃圾回收中,为了保证每个数据块在垃圾回收后只对应一个转换页,本文方法将基于转换页数据聚集技术复用在垃圾回收中.在地址映射中,数据请求的发起者为文件系统,而在垃圾回收中为垃圾回收器,两者具有相似的地址映射过程,因此方法可以复用.在数据块的垃圾回收过程中,当垃圾回收器需要拷贝一个有效页时,首先访问全局转换页映射以获得当前的可用数据页的物理地址(APN),如果可用数据页是存在的(非−1),则直接将有效页写入可用数据页;否则,如果当前分配的数据块已经用完,则使用交换块(swap block)作为新分配的数据块存储有效页.当垃圾回收完成时,擦除后的空数据块作为新的交换块.

图 5 为数据块垃圾回收的示例.选中的回收数据块中有 4 个有效数据页(Q,B,D,E),当拷贝有效页 Q 时,方法首先定位到对应的 GTD 项,获得 APN 为 68,因此,Q 可以直接拷贝到数据块 23 的数据页 68 中;拷贝后,APN 从 68 更新到 69.同理,对于数据 B 也可以拷贝到数据页 69 中.此时,数据块 23 已经写满,所以 APN 被置为−1.当下一个数据拷贝请求 D 到来时,分配交换块给 GTD 项,因此,D 和 E 都写入交换块中,APN 也更新为交换块中下一个可用的数据页地址.最终,数据块 23 擦除后将成为新的交换块,为下一次垃圾回收服务.

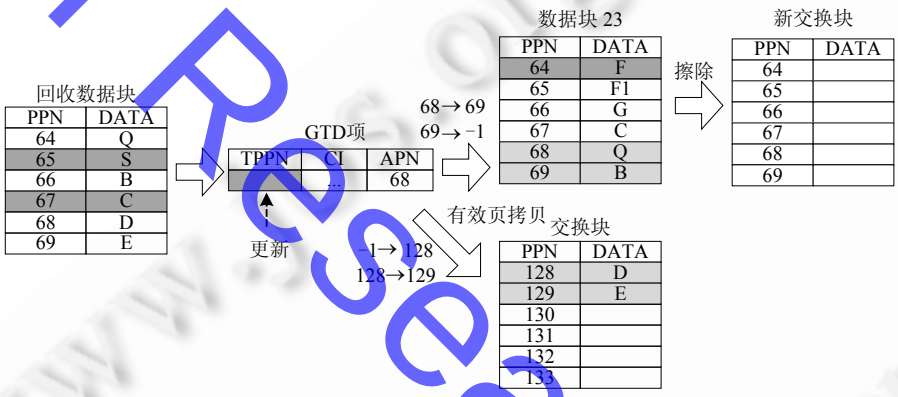


Fig.5 Translation based data block garbage collection

图 5 基于转换页的数据块垃圾回收

对地址转换块的垃圾回收,闪存中维护一个当前可用的地址转换页,相当于统一的写指针,因此,转换页将被按顺序写入到当前地址转换块中.当闪存中无可分配的地址转换块时将触发垃圾回收,当垃圾回收时,回收块中的有效转换页将拷贝到写指针指向的可用地址转换页.同时,如果地址转换页位于缓存,则直接从缓存写入当前转换块,减少了对有效转换页的读操作和将来缓存产生的转换页更新操作.最后,页级地址映射缓存的置换将相近热度的转换页写回相同的转换块,间接地降低垃圾回收时转换块中的有效页的数目,从而降低了转换页的更新开销.图 6 所示为地址转换块的垃圾回收,选中回收的地址转换块包含 TP0,TP2,TP4 这 3 个地址转换页,这 3 个地址转换页被拷贝到当前地址转换块中(TPPN:146~148).拷贝过程中,如果存在缓存,则直接从缓存写入当前地址转换块,拷贝后对应的 GTD 也同时更新,以确保能够定位到最新的地址转换页.

为了平衡数据块的擦写次数延长闪存的耐久性,闪存中记录了每个数据块的擦写次数,全局转换页映射表中能够维护数据区域的热度信息,记录每个地址转换页对应连续逻辑空间的访问频率,对频繁访问的项在写指针请求分配时选择擦写次数少的“冷”块,而对不频繁访问的则分配擦写次数较多的“热”块.同样,在垃圾回收中,选择擦写次数较少且无效页较多的数据块作为回收块进行擦写.在闪存空闲时间里,可对冷、热数据块进行数据交换以平衡块间擦写次数.

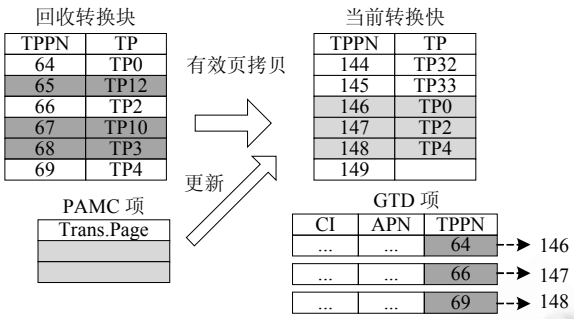


Fig.6 Translation block garbage collection

图 6 转换块的垃圾回收

3 实验与分析

在实验中,使用一系列的基准数据集仿真评估本文方法的有效性.本文将提出的方法(简称 OAT)和一种代表性的基于需求的页级映射方法(DFTL)进行了实验比较.实验中使用 4 个性能度量指标:缓存命中率、转换页操作数、系统响应时间和块擦除次数.

3.1 实验框架

图 7 为实验仿真平台框架.其中,DiskMon[19]为磁盘驱动数据跟踪器,收集访问磁盘的 I/O 数据请求;DiskSim[20]是一个广泛用于工业界的成熟磁盘仿真框架;FlashSim[15]作为 DiskSim 的一个扩展组件,用于仿真闪存芯片的管理方法和基本操作.在仿真实验中,本文在 FlashSim 中实现了 DFTL 和本文的地址映射方法,配置了一个 32GB 的 NAND 闪存存储系统.具体参数见表 2.

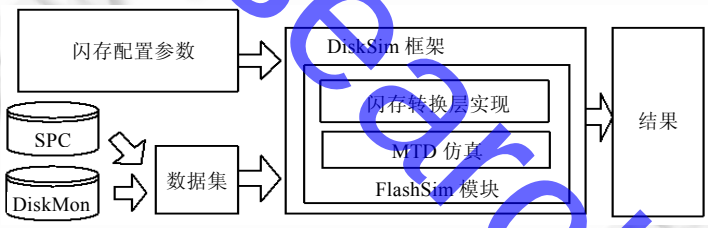


Fig.7 Simulation framework in experiment

图 7 实验中使用的仿真框架

Table 2 Parameters of the flash memory in experiment

表 2 实验中的闪存参数

参数	配置值
总容量(GB)	32
预留空块比例(%)	15
最小空块数	3
块数量	8×2048
每块页数	64
页大小(KB)	2
写页延迟(ms)	0.029
读页延迟(ms)	0.205 9
块擦写延迟(ms)	1.5

实验采用实际 I/O 数据请求研究不同地址映射方法对性能的影响.I/O 数据集的基本情况见表 3.其中,Websearch[21]来自 SPC(storage performance council),是一个读为主的 I/O 数据集.Financial 来自 OLTP 应用[21],

是一个连续访问的数据集,其系统运行在商业数据中心.该数据集对应的逻辑空间较小.Systemdisk1~Systemdisk3 为通过 Diskmon 收集的 PC 上的磁盘访问请求.其中,Systemdisk1 数据集环境为多个文件的拷贝操作,具有较高的写操作比例.Systemdisk2 为视频在线播放环境下收集的数据集,因此具有较高的请求数,连续写操作较多,但是请求长度较小.Systemdisk3 为多程序数据处理,随机写操作较多,相邻写操作间的地址距离较大.因此,实验数据集覆盖了企业级的读为主和写为主的数据环境,也包括了日常数据环境下的连续写、随机写、不同请求长度的各类数据集,具有一定的代表性.实验前,通过预处理过程将数据集中的读请求预先写入仿真的 NAND 闪存.

Table 3 Trace used in the simulation

表 3 仿真数据集

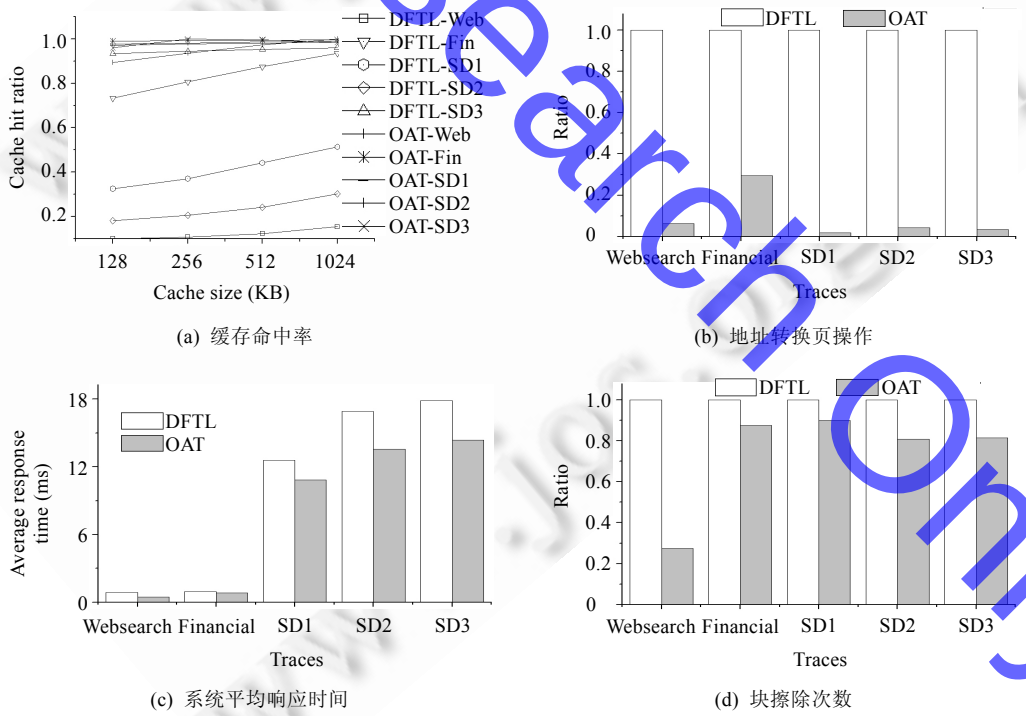
数据集	请求数	写操作比例(%)	平均请求长度(KB)
Websearch	1 055 448	0.02	15.05
Financial	3 698 864	17.66	5.24
Systemdisk1	670 412	71.89	42.30
Systemdisk2	1 730 415	67.21	41.10
Systemdisk3	875 928	63.44	47.75

3.2 实验结果

实验比较了 DFTL 和本文提出的地址映射方法,根据 4 个性能度量指标——缓存命中率、转换页的操作数、系统响应时间和块擦除次数,具体分析和讨论实验结果.

3.2.1 缓存命中率

缓存命中率是影响系统性能的重要因素之一.为公平比较,实验评估了同样的内存开销下,本文方法和 DFTL 的缓存命中率.地址映射缓存的大小设定为 128KB,256KB,512KB 和 1 024KB.实验结果如图 8(a)所示,本文方法可达到至少 89.72%的缓存命中率,而 DFTL 只有平均 40.14%的缓存命中率.



(a) 缓存命中率 (b) 地址转换页操作

(c) 系统平均响应时间 (d) 块擦除次数

Fig.8 Experimental results

图 8 实验结果

在基于需求的地址映射方法中,每个缓存页可以保存 1MB(512×2KB)地址空间的映射信息.从实验结果可以看出,缓存大小增长可以显著提高缓存的命中率.其中,对读为主的数据环境,由于读请求的地址较为随机,缓存空间增加的情况下大约只有 5%的命中率提升.而写为主的数据环境中有大量的连续写操作,缓存命中率在缓存增加时大约有 12%~20%的提升.Systemdisk1,Systemdisk2 为连续写为主的环境,而 Financial 数据集中请求的数据逻辑地址的距离普遍偏小,因此,相对于 Systemsdik3 的随机写有更高的缓存命中率.为了相对公平地进行实验比较,对于其他性能度量指标,选用 512KB 作为缓存大小进行后续的实验并分析和讨论实验结果

3.2.2 转换页操作

本文方法采用页级地址映射缓存,以转换页为单位进行缓存,提高了缓存命中率.从缓存命中率的实验结果可以看出,本文方法可以提高 20%左右的缓存命中率.因此,缓存未命中造成的额外转换页读写操作显著减少,并且多个地址映射更新信息可以批量写入转换页.通过采用基于转换页指导的数据聚集技术,本文方法将每个数据块对应的转换页数量降至最低的常量值,因此当数据块进行垃圾回收时,最多只需对 1 个转换页更新,与 DFTL 相比,显著减少了转换页的更新操作.如图 8(b)所示,为方便比较,将 DFTL 的数据进行归一化处理,从实验结果可以看到,无论读为主还是写为主的数据环境,本文方法在转换页操作上均有显著的降低,平均减少 90.93%的转换页操作.Financial 作为地址空间范围较小的数据集,每个数据块中对应不同地址映射页的数量较少,因此,在垃圾回收机制中,对应地址映射页更新数量的降低与其他数据环境相比并不显著.对于读为主的数据环境,由于缓存命中率的提升,转换页的操作大量减少;对于写为主的数据环境,在随机写居多的 Systemdisk3 能够获得更显著的降低效果.

3.2.3 系统平均响应时间

系统平均响应时间是闪存存储系统性能的重要评估指标之一,实验比较了 DFTL 和本文方法的系统平均响应时间.影响系统平均响应时间的关键因素是垃圾回收机制,因此实验主要评估了垃圾回收运行情况下对系统平均响应时间的影响.实验结果如图 8(c)所示,本文方法平均提高了 22.14%的系统平均响应时间.在读为主的数据集下,由于写操作较少,垃圾回收来源于地址转换页的更新导致的转换块的垃圾回收,由于转换块垃圾回收触发次数较少,因此对系统平均性能影响较小.而 Financial 数据集地址空间范围较小,触发垃圾回收时,数据块中存在大量的无效页,有效页拷贝较少,所以对系统平均响应时间影响不大,本文的方法只有略微的提高.对写为主的数据环境,由于有效页拷贝较多,所以整体的响应时间大于前两者,对于随机写环境下的 Systemdisk3 数据集,由于优化了垃圾回收时转换页的更新,降低了额外的读写操作,能够显著提升系统平均响应时间.

3.2.4 块擦除次数

从第 3.2.2 节可以看到,本文方法显著减少了转换页的读写操作,因此属于同一转换页的地址映射信息能够在缓存置换时集中更新,减少了闪存转换页的消耗,从而减少了转换块的擦除次数.另一方面,通过基于转换页的数据聚集技术,数据块中的数据分布得到了优化,相近逻辑地址的数据被存储在同一数据块中,利用空间局部性,这些数据页很可能被集中更新,因此减少了回收块中的有效页数目,从而获得更多的空数据页及数据块,间接降低了数据块的擦除次数.如图 8(d)所示,与 DFTL 相比,本文方法平均减少了 26.51%的块擦除次数.由于转换页操作的减少能够显著降低转换块的垃圾回收次数,因此在读为主的 Webserarch 数据集下,本文方法能够显著降低块擦写次数;而对写为主的其他数据集,本文方法也能获得不同程度块擦除次数的优化,特别是对于随机写的数据环境,本文方法使用数据聚集技术将逻辑地址相近的数据组织到同一个数据块中,利用空间局部性,大部分相近逻辑地址的数据更容易一起更新,因此显著降低了块擦除的次数.

3.3 系统开销

为了优化地址映射的方法,减少转换页操作带来的性能开销,一方面,本文方法扩展了全局转换页映射表(GTD),在表中每一项记录了缓存索引(CI)和可用数据页地址(APN),增加了内存开销.然而,GTD 的内存开销本身非常小,对于 1GB 闪存只消耗大约 2KB 的内存空间.在本文方法中,对于 32GB 的 NAND 闪存,GTD 的开销是 96KB,能够符合一般嵌入式系统的内存要求.另一方面,由于采用了基于转换页的数据聚集技术,不常访问的冷数据可能长期占用某些数据块,从而降低了闪存的空间利用率.在实验中,大约有 4.74%的闪存空间开销.虽然有

部分内存和闪存空间的开销,但是本文方法比现有的方法减少了平均90.93%的转换页操作,并提升了22.14%的平均系统性能.

4 结 论

本文提出了一种基于需求的页级闪存地址映射优化方法.一方面,设计页级地址映射缓存统一内存和闪存的映射信息粒度,利用其空间局部性提高缓存命中率,同时,融合了全局转换页映射表和页级地址映射缓存两种数据结构以优化缓存访问;另一方面,通过设计基于转换页的数据聚集技术,将地址映射信息属于同一转换页的数据写入相同的数据块,降低了垃圾回收时转换页的更新数目.实验中,使用一系列基准数据集仿真评估本文方法,结果显示,文本方法能够显著减少转换页操作并提升了系统性能.

下一步,本文方法将应用于固态硬盘(SSD),在多闪存芯片并行环境中优化地址映射;同时,针对内存极其有限的嵌入式环境,研究动态多粒度的地址映射方法.

References:

[1] Chung TS, Park DJ, Park S, Lee DH, Lee SW, Song HJ. A survey of flash translation layer. Journal of Systems Architecture, 2009, 55(5-6):332−343. [doi: 10.1016/j.sysarc.2009.03.005]

[2] Ma DZ, Feng JH, Li GL. LazyFTL: A page-level flash translation layer optimized for NAND flash memory. In: Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2011. 1−12. [doi: 10.1145/1989323.1989325]

[3] Gupta A, Kim Y, Urgaonkar B. DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings. In: Proc. of the 14th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. New York: ACM Press, 2009. 229−240. [doi: 10.1145/1508244.1508271]

[4] Qin ZW, Wang Y, Liu D, Shao ZL. A two-level caching mechanism for demand-based page-level address mapping in NAND flash memory storage systems. In: Proc. of the 2011 17th IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS). IEEE, 2011. 157−166. [doi: 10.1109/RTAS.2011.23]

[5] Choudhuri S, Givargis T. Performance improvement of block based NAND flash translation layer. In: Proc. of the 5th IEEE/ACM Int'l Conf. on Hardware/Software Codesign and System Synthesis. New York: ACM Press, 2007. 257−262. [doi: 10.1145/1289816.1289878]

[6] Choudhuri S, Givargis T. Deterministic service guarantees for NAND flash using partial block cleaning. In: Proc. of the 6th Int'l Conf. on Hardware/Software Codesign and System Synthesis. New York: ACM Press, 2008. 19−24. [doi: 10.1145/1450135.1450141]

[7] Qin Z, Wang Y, Liu D, Shao Z. Demand-Based block-level address mapping in large-scale NAND flash storage systems. In: Proc. of the 8th IEEE/ACM Int'l Conf. on Hardware/Software Codesign and System Synthesis. New York: ACM Press, 2010. 173−182. [doi: 10.1145/1878961.1878991]

[8] Chang YH, Kuo TW. A commitment-based management strategy for the performance and reliability enhancement of flash-memory storage systems. In: Proc. of the 46th Design Automation Conf. New York: ACM Press, 2009. 858−863. [doi: 10.1145/1629911.1630130]

[9] Cho H, Shin D, Eom YI. KAST: K-Associative sector translation for NAND flash memory in real-time systems. In: Proc. of the Design, Automation & Test in Europe Conf. & Exhibition. IEEE, 2009. 507−512. [doi: 10.1109/DATE.2009.5090717]

[10] Qin Z, Wang Y, Liu D, Shao Z, Guan Y. MNFTL: An efficient flash translation layer for MLC NAND flash memory storage systems. In: Proc. of the 48th Design Automation Conf. New York: ACM Press, 2011. 17−22. [doi: 10.1145/2024724.2024730]

[11] Wu CH, Kuo TW. An adaptive two-level management for the flash translation layer in embedded systems. In: Proc. of the IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD 2006). IEEE, 2006. 601−606. [doi: 10.1109/ICCAD.2006.320107]

[12] Qi XY, Tang X, Liang ZC, Meng XF. OAFTL: An efficient flash translation layer for enterprise application. Journal of Computer Research and Development, 2011,48(10):1918−1926 (in Chinese with English abstract).

[13] Shi L, Li J, Xue CJ, Yang C, Zhou X. Exlru: A unified write buffer cache management for flash memory. In: Proc. of the 2011 Int'l Conf. on Embedded Software (EMSOFT). IEEE, 2011. 339−348. [doi: 10.1145/2038642.2038694]

[14] Shi L, Xue CJ, Zhou X. Cooperating write buffer cache and virtual memory management for flash memory based systems. In: Proc. of the 2011 17th IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS). IEEE, 2011. 147−156. [doi: 10.1109/RTAS.2011.22]

[15] Zhang Q, Li X, Wang L, Zhang T, Wang Y, Shao Z. Optimizing translation information management in NAND flash memory storage systems. In: Proc. of the 2013 18th Asia and South Pacific Design Automation Conf. (ASP-DAC). IEEE, 2013. 326−331. [doi: 10.1109/ASPDAC.2013.6509616]

[16] A simulator for various FTL schemes. Department of Computer Science and Engineering, The Pennsylvania State University, 2009. http://csl.cse.psu.edu/?q=node/322

[17] Corporation S. Samsung electronics. Samsung K9G4G08U0A (v1.0)-4GB MLC NAND flash data sheet. 2006.

[18] Kuo TW, Chang YH, Huang PC, Chang CW. Special issues in flash. In: Proc. of the IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD 2008). San Jose: IEEE, 2008. 821−826. [doi: 10.1109/ICCAD.2008.4694174]

[19] Russinovich M. DiskMon for Windows v2.01. 2006. http://technet.microsoft.com/enus/sysinternals/bb896646.aspx

[20] Bucy JS, Ganger GR. The DiskSim simulation environment version 3.0 reference manual. 2003. http://www.pdl.cmu.edu/DiskSim/

[21] Websearch trace and OLTP trace from umass trace repository. 2012. http://traces.cs.umass.edu/index.php/Storage/Storage

附中文参考文献:

[12] 綦晓颖,汤显,梁智超,孟小峰.OAFTL:一种面向企业级应用的高效闪存转换层处理策略.计算机研究与发展,2011,48(10): 1918−1926.

张琦(1987－),男,安徽芜湖人,博士生,主要研究领域为闪存存储系统,嵌入式系统,实时系统.
E-mail: zhangqi@seg.nju.edu.cn

张天(1978－),男,博士,副教授,CCF 高级会员,主要研究领域为模型驱动工程.
E-mail: ztluck@nju.edu.cn

王林章(1973－),男,博士,副教授,CCF 高级会员,主要研究领域为模型驱动的软件测试与验证,软件测试自动化.
E-mail: lzwang@nju.edu.cn

邵子立(1973－),男,博士,副教授,CCF 会员,主要研究领域为嵌入式软件和系统,实时系统.
E-mail: cszlshao@comp.polyu.edu.hk