文章编号:1671-4598(2018)08-0270-05 DOI:10.16526/j.cnki.11—4762/tp.2018.08.060 中图分类号:TP242 文献标识码:A

基于闪存特性的 EXT4 文件系统读写性能优化研究

陶琛嵘，陈莉君
（西安邮电大学 计算机学院，西安 710061）

摘要：Android 智能手机普遍采用闪存作为本地存储介质，这导致使用的 EXT4 文件系统存在读写性能损耗问题，间接影响了应用程序的使用体验，在分析性能损耗原因的基础上，提出一种针对 EXT4 文件系统日志功能优化方法；首先对 EXT4 文件系统日志功能进行研究，分析其如何维护日志数据，对比传统磁盘及闪存的不同特性提出性能损耗问题产生的原因，并给出解决方法，优化 EXT4 日志数据存取流程，以 EXT4 事务状态作为依据主动删除无用的日志数据，在不破坏原有日志功能的基础上，减少闪存设备垃圾回收过程中不必要的数据拷贝；通过对比实验证明，该方法能够提升 Android 手机中 Ext4 文件系统约 11.8%的读写性能。

关键词：EXT4；日志；闪存设备；垃圾回收；Android 系统

Research on Performance Optimization of EXT4 File System Based on Flash Memory

Tao Chenrong, Chen Lijun
(Xi'an University of Posts and Telecommunications, Xi'an 710061, China)

Abstract: Android mobile phones generally use flash as a local storage medium, which leads to the use of EXT4 file system has the problem of loss of read and write performance, indirectly affecting the application experience. Based on the analysis of the causes of performance loss, this paper presents a method for optimizing the log function of EXT4 file system. First of all, EXT4 file system log function research, analysis of how to maintain journal data. By comparing the different characteristics of traditional disk and flash memory, we put forward the reasons for the performance loss problem and give the solution. This method takes EXT4 transaction status as a basis to actively delete useless log data, and reduces unnecessary data copy in flash device garbage collection process without destroying the original journal function. Through the comparative experiments show that this method can improve about 11.8% read and write performance of EXT4 file system in Android mobile phone.

Keywords: EXT4; journal; flash memory; garbage collection; Android system

0 引言

Android 是一款基于 Linux 内核的开源操作系统，一经提出便在移动设备上迅速普及，目前 Android 系统已经占据了全球智能手机五成份额[1]。Android 智能手机广泛使用体积小、功耗低的 eMMC（Embedded Multi Media Card）作为数据存储设备，区别于传统机械硬盘，eMMC 使用闪存芯片作为存储介质[2]。同时因手机电源不稳定的特点，Android 默认使用 EXT4 文件系统[3]，作为 Linux 中最常用的日志文件系统，采用 JBD2（Journal Block Device 2）软件[4]进行日志记录，因此拥有良好的故障恢复能力，能够保证 Android 系统的稳定性以及数据的安全性。目前 Android 存储系统架构与 Linux 存储系统架构类似，有着用户空间与内核空间两部分。内核空间中主要包括 EXT4 文件系统与 eMMC 存储设备两个部分，EXT4 文件系统通过虚拟文件系统向用户空间提供系统调用接口，而 EXT4 文件系统底层则通过通用块层、块设备驱动层与 eMMC 块设备关联起来[5]。

目前针对 Android 系统存储性能优化的方案主要有两个方面：一方面是用户空间中对应用软件或者软件依赖运行库的优化；另一方面是内核空间中对文件系统的优化。用户空间中的优化主要集中于对 Android 中大量使用的 SQLite 数据库技术的改进，例如通过消除 SQLite 的 WAL 模式下数据库 checkpoints 的写冗余部分，获取存储性能的提升[6]，但是这样的方案只能够提升软件利用 SQLite 技术

收稿日期:2018-01-22； **修回日期**:2018-02-22。

作者简介:陶琛嵘(1993-),男,硕士研究生,主要从事嵌入式系统研究与开发方向的研究。

陈莉君(1964-),女,教授,硕士研究生导师,主要从事 Linux 内核方向方向的研究。

存储数据时的存储效率，无法对 Android 系统中文本文件、多媒体文件等多种类型的文件读写产生有效提升。内核空间中的优化主要包括对 Android 默认使用的 EXT4 文件系统的优化以及针对 Android 使用的闪存设备设计专门的文件系统两种方案，前者优化调整 EXT4 文件系统中的重要参数从而提升文件系统的整体存储性能，但是方案中并没有考虑到 EXT4 文件系统的日志功能用于闪存设备所造成的影响，而且效果并不理想[7]；后者则有着著名的 F2FS（Flash Friendly File System）[8]，此方案虽然在小文件的随机读写上有着较大的提升，但是由于开发时间尚短，在系统稳定性、软件兼容性等方面仍无法与 EXT4 文件系统相比，因此目前尚未广泛使用。

针对 Android 智能手机普遍采用的 eMMC 存储设备挂载 EXT4 文件系统的存储系统架构，本文分析了 EXT4 文件系统日志特性导致其对闪存设备文件读写性能衰减的问题，提出了一种能够消除此问题的 EXT4 文件系统日志功能优化方案。

1　相关理论

1.1　EXT4 日志文件系统

EXT4 文件系统借用了数据库中事务的思想，为需同步至磁盘的数据建立相关的日志数据，以便能够在发生系统崩溃后可以利用日志数据恢复，重新使文件系统保持一致的状态[9]。EXT4 文件系统挂载时将磁盘空间划出一小块空间作为日志数据区域，专门用于存储日志数据。文件系统读写文件时，保持原有的读写磁盘逻辑不变，通过 JBD2 进程将影响文件系统一致性的元数据块及时写入到日志数据区域中。

表 1　EXT4 文件系统数据同步事务状态表

状态	描述
Running	该事务依旧存活并且可以接受更多的操作
Locked	该事务不能够再接受新的操作，但是已经接受的操作仍未完成
Flushed	该事务数据正在写入到日志中
Committed	该事务数据已经写入到日志区域中并且对文件系统的改变正在同步中
Finished	该事务数据已经被全部写入到日志数据区域以及通用数据区域中，事务完成

图 1 描述了 EXT4 文件系统日志数据流。虚线箭头表示了当文件系统读写数据中途崩溃时的修复过程，重新挂载分区时根据日志数据区域中的数据，逐一将记录下的元数据写回到磁盘原始位置，保证文件系统的一致性。实线箭头表示每次 EXT4 文件系统读写磁盘时的逻辑，首先由 JBD2 进程将每次需要写入的数据中元数据部分拷贝一份，将元数据封装成日志数据后提交到日志数据区域，之后 EXT4 文件系统对磁盘进行正常读写数据流程，这样的一次流程被称为一次事务。表 1 为一个事务几种状态的描述[10]，以区分事务的执行程度。

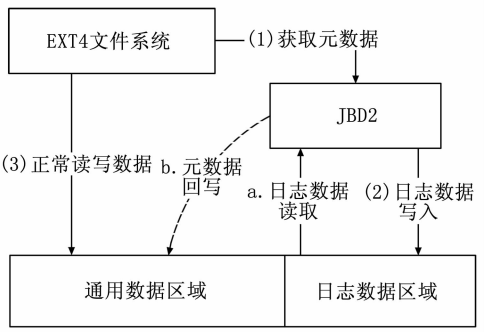

图 1　EXT4 文件系统日志数据流图

EXT4 的日志数据区域以文件系统的块为单位组织日志数据，有着类似于通用数据区域的结构划分，但是相比通用数据区域更加简单，起始位置存放了日志数据的超级块，用于组织管理日志数据区域，随后便是按顺序以及一定规则存放的各次事务日志数据。一次事务的日志数据主要由描述块、数据块和提交块组成，其中描述块标识了一份日志数据的起始位置，数据块则存储了日志数据内容，提交块用于表明本次事务是一次完整的事务，在文件系统崩溃恢复时只会使用完整的事务数据。在进行文件系统崩溃恢复时，需要找到日志数据的数据块所在原文件系统的块地址进行回写修复，因此日志数据中的各个数据块与原文件系统的目标块的对应关系就尤为重要，这种对应关系存放在描述块中。由于数据块数量不定，JBD2 在描述块中通过一个数据结构表明一组对应关系，该数据结构为：

```
typedef struct journal_block_tag_s
{
    __be32    t_blocknr;
    __be32    t_flags;
    __be32    t_blocknr_high;
} journal_block_tag_t;
```

该结构体通过其在描述块中序号标识代表的数据块号，通过 t _ blocknr 字段标识对应的原文件系统块地址。每个描述块中都包含了一个或多个 journal _ block _ tag _ t 结构体用于表明本次事务日志数据中各数据块对应的文件系统原始位置。

1.2　闪存转换层

闪存转换层（Flash Translation Layer，简称 FTL）是一种通过块设备模拟方式实现闪存设备存储系统的技术。将底层 NAND 闪存介质的管理及操作封装起来，为文件系统提供块设备读写接口，从而让文件系统访问 NAND 型闪存就像访问传统机械硬盘一样[11]。闪存转换层的功能主要有两点：地址映射及垃圾回收。一般情况下，闪存转换层通过维护两个静态表来完成它的功能[12]，如图 2 所示。

闪存转换层通过其内部维护的地址映射表和空间管理表来管理逻辑地址转换以及闪存的物理空间。当文件系统通过块设备读写接口读写块设备时，由闪存设备接收对逻

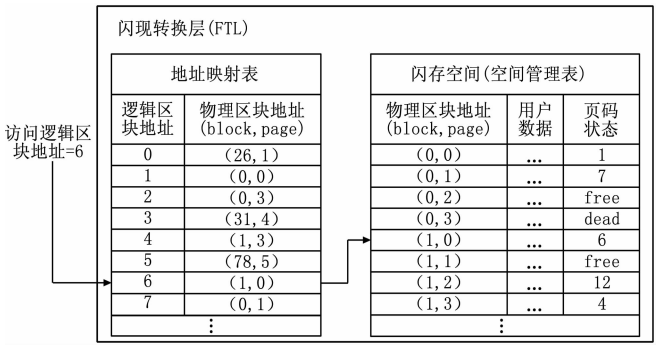

图 2 闪存转换层结构图

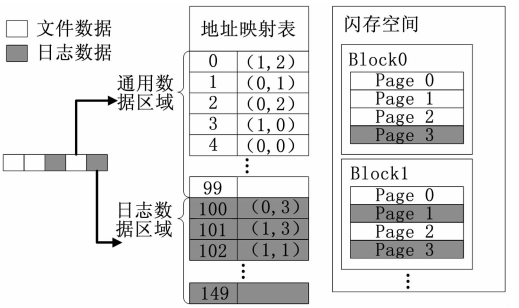

图 3 通用数据及日志数据存储状况

辑区块地址的读写请求，根据读写请求中的逻辑区块地址，在闪存转换层中通过地址映射表找到对应的物理区块地址进行访问，这就是闪存转换层的地址映射功能。垃圾回收即对闪存中的无用数据进行回收，闪存转换层指定一个闪存块进行垃圾回收，通过空间管理表中的页面状态判断当前页中的数据是否需要保留，对需要保留的数据页（Live，页面状态中记录了物理页面对应的逻辑区块地址）暂时拷贝至缓存或其他区域，等待块擦除完成再次写入原位置，对于处于已经无用的数据页面（页面状态为 dead）则无需拷贝，直接擦除[13]。

2 EXT4 文件系统闪存设备的性能衰减分析

EXT4 文件系统挂载时对块设备划分的日志数据区域是对逻辑区块地址的划分，即针对闪存转换层维护的地址映射表中逻辑区块地址进行了通用数据区域与日志数据区域的划分。但是当实际数据同步至闪存设备时，逻辑区块地址实际映射的闪存物理地址是根据闪存设备自身损耗均衡等机制确定并建立映射的，这样就会导致在 EXT4 文件系统提出 IO 请求的逻辑区块地址上，通用数据与日志数据严格分离，但是在闪存设备中，两种数据却可能是混杂在同一个闪存块中的。

如图 3 所示，日志数据区域是逻辑区块地址范围为 100 至 149 的区域，当 EXT4 文件系统开始将数据同步至闪存设备时，文件数据写入到通用数据区域，而日志数据写入到日志数据区域，两个区域都是连续且严格分离的，但是数据写入到闪存设备具体页时，数据会由闪存设备根据自身的策略选择一个空白页进行写入，这就导致了在一个闪存设备的物理块中，日志数据的页与通用数据的页是混杂的。

EXT4 文件系统通过 JBD2 将日志数据提交到日志数据区域，JBD2 软件通过函数 jbd2 _ journal _ next _ lob _ block 函数获取下一个可写入的逻辑区块地址，此函数的实现决定了日志数据以何种顺序记录在日志数据区域中。jbd2 _ journal _ next _ log _ block 函数中由 j _ head 变量作为日志数据区域的写指针，用于指向下一个可写的逻辑区块地址，在每次写入一个逻辑块后通过自增的方式改变写指针位置。当指针指向日志数据区域尾部即日志数据区域全部写满后，指针会重新指向日志数据区域头部，即日志数据再次从区域头部开始写入。采用这样方式能够利用传统机械硬盘的可覆写特性，不对以往写入的日志数据执行删除操作，直接覆盖写入新的日志数据。如图 4 所示。

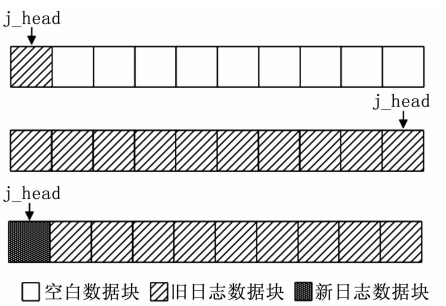

图 4 JBD2 日志数据区域循环结构原理

根据以上情况可以看出，EXT4 文件系统可以在日志数据区域不断的循环写入日志数据，在满足日志数据写指针到达日志数据区域尾部之前，日志数据不会主动删除，对于闪存设备而言，这些数据随机分散在闪存设备的各个块中，并且与通用数据混杂在一起，闪存设备进行垃圾回收时，这些数据所在页的状态仍为 Live，导致闪存设备垃圾回收时仍然需要拷贝这部分无用日志数据，造成文件系统读写闪存文件性能衰减的现象，本文对此问题提出相应解决方案。

3 优化方案

本文针对以上描述的问题，提出主动删除事务日志数据策略，根据 EXT4 文件系统数据同步事务处于 Finished 状态时，通用数据以及日志数据已经全部写入磁盘的含义，结合 JBD2 日志数据区域循环结构的特性，在事务状态变为 Finished 时记录本次事务日志数据的结束逻辑区块地址，多次数据同步事务的过程中，通过上次事务与本次事务保存下来的逻辑区块地址确定本次事务的日志数据位置，进行主动删除操作。对于开启日志功能的 EXT4 文件系统，优化后单次数据同步操作的基本流程如图 5 所示。

通过多次数据同步事务中对图 5 中描述的流程不断循环，达到及时删除无用日志数据的目的，消除因日志功能导致的 EXT4 文件系统读写性能衰减问题。根据事务的 Finished 状态记录下本次事务日志数据的结束逻辑区块地

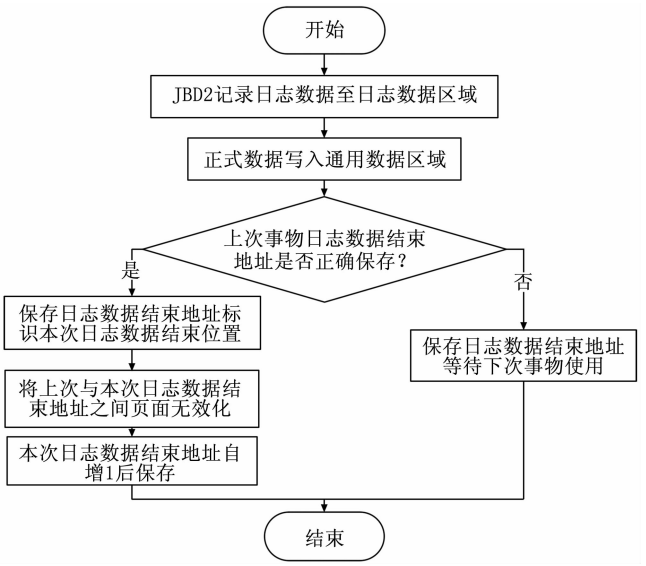

图 5　优化后 EXT4 单次数据同步事务流程

址，同时检查是否存在有效的上次事务日志数据的结束逻辑区块地址，若存在，则两地址之间的页保存了本次事务的日志数据，将这段逻辑区域地址对应的闪存物理页面无效化，从而将日志数据删除；若不存在，则不进行闪存页面无效化操作，本次事务保存的地址在下次事务时标记了日志数据的起始区块地址，从而实现下次事务的日志数据删除操作。

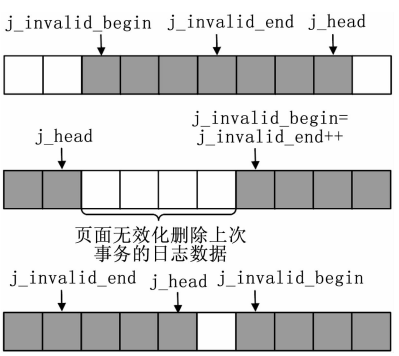

图 6　主动删除事务日志数据策略原理

如图 6 所示，在日志数据区域中，事务日志数据的主动删除策略通过 j _ invalid _ begin 与 j _ invalid _ end 两个变量记录事务日志数据的逻辑区块地址，将日志数据删除，完成后将 j _ invalid _ end 向后移动，作为下次删除操作的起始位置。当日志数据的删除指针到达日志数据区域尾部时，采取与日志循环写入结构相同的策略，删除指针重新指向日志数据区域头部。

本文为 EXT4 文件系统的日志功能实现了一个额外的循环结构，将不会再被使用到的日志数据删除，无用的日志数据基于事务 Finished 状态进行删除，Finished 状态即表示正式数据已经写入完毕，不再需要该事务的日志数据预防宕机、EXT4 文件系统崩溃等情况，因此不会出现有用的日志数据被删除的情况，保证系统稳定性的同时提高文件系统读写性能。

4　实验分析

4.1　实验环境

为验证本文提出的 EXT4 文件系统日志功能优化方案，进行实机测试比较，所用机型为 Samsung Galaxy S5，具体环境参数如表 2 所示。

表 2　测试环境

软、硬件	参数值
CPU	高通骁龙 801
Memory	2GB
Android 系统版本	Android 4.4.2
Linux 内核版本	Linux 3.4.0

对 Linux 内核修改后编译并移植至测试机器中，进行相应测试。

4.2　测试分析

首先通过 A1 SD Bench 进行测试，这是一款 Android 手机存储性能测试工具，能够测试手机内置闪存、RAM 等设备的性能。执行 50 次 A1 SD Bench 闪存性能测试，每次分别进行 100 秒读取及写入操作，统计读取与写入的数据量计算出对闪存读取速度与写入速度。如图 7、8 所示，图中横坐标为测试次数序号，纵坐标为读、写速度。

通过图 7 可以看到，灰色线条表示优化前读取速度折线图，黑色线条则表示优化后读取速度。优化前 EXT4 读取速度范围约为 166.6 Mb/s 至 181.3 Mb/s，而优化后读取速度范围约为 187.4 Mb/s 至 199.8 Mb/s，虽然速度存在一定波动，但是从图中可以看出，优化后的读取速度相比优化前有着明显的提升。

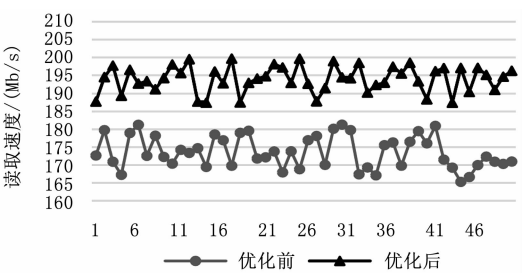

图 7　优化前后读取速度比较

通过图 8 可以看到，与读取速度类似，写入速度也存在波动的现象，优化前的 EXT4 写入速度范围约为 20.2 Mb/s 至 22.2 Mb/s，优化后的写入速度范围约为 22.6 Mb/s 至 24.63 Mb/s。虽然优化前写入速度最大值与优化后写入速度的最小值之间的差距并不大，甚至在第 18 次的测试中，优化前后的写入速度差仅为 0.49 Mb/s，但是在整体折线图的分布上，优化后写入性能有着明显提升。此外通过以上测试可以看出，读取速度明显优于写入速度，这主要是由于 Android 所使用 NAND 闪存介质本身读

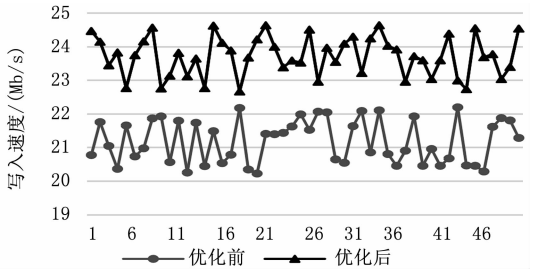

图 8 优化前后写入速度比较

取性能远高于写入性能导致的。对以上测试数据进行统计计算后，本方案优化后文件系统的读写性能提升约 11.8%。接下来编写程序测试优化前后 EXT4 对不同大小文件的读写时间，通过 O_DIRECT 标志绕过文件系统缓存，对各种大小的文件分别测试 10 次统计平均值，测试结果如图 9 所示。

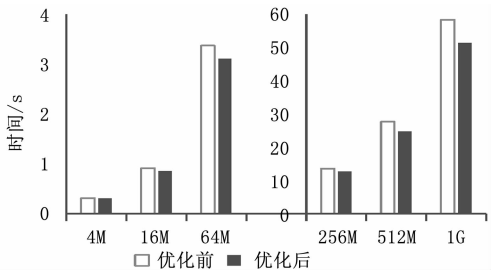

图 9 不同文件大小读写时间测试

如图 9 所示，纵坐标代表不同大小文件的读写速度，修改后 EXT4 对相同大小文件的读写花费的时间相比于原内核更短。在文件较小时，优化前后花费的时间差别较小；但是当文件较大时，优化后所花费的时间有着明显的减少。这是因为对于小文件来说，可能触发的垃圾回收过程较少甚至不需要进行垃圾回收，因此本方案优化效果不明显，但是在进行大文件的读写时，会触发更多的闪存垃圾回收，本方案能够减少闪存垃圾回收中对无用日志数据的拷贝消耗，因此提升效果较为明显。

5 结论

本文提出一种提高 EXT4 文件系统性能优化方案，首先记录数据同步事务处于 Finished 状态时日志数据的逻辑区块地址，结合上一事务记录下的逻辑区块地址，令本次日志数据所在闪存物理页面无效，最后在垃圾回收时清除闪存设备的无用日志数据，从而避免日志数据的冗余拷贝。通过实验对比，本文方法能够减轻闪存设备因日志数据造成的“写放大”现象，同时能够在增强 EXT4 文件系统读写效率的同时，延长闪存设备的使用寿命。

在本文研究基础上，下一步研究工作主要有以下几个方面，考虑除日志数据以外导致 Android 存储性能下降的其他因素，例如针对 SQLite 与 EXT4 的日志记录不协调问题进行优化研究等，实现 Android I/O 栈从应用程序至底层硬件设备的一整套优化方案，进一步提高 Android 智能手机的存储性能。

参考文献：

[1] 贡知洲，路昭亮. Android 发展的分析与研究 [J]. 价值工程，2013，32 (2)：185－186.

[2] 王举利. eMMC 存储系统的闪存转换层研究与设计 [D]. 天津：天津工业大学，2015.

[3] Kim D，Park J，Lee K G，et al. Forensic Analysis of Android Phone Using Ext4 File System Journal Log [J]. 2012，164：435－446.

[4] Mathur A，Cao M，Bhattacharya S，et al. The new EXT4 file system：Current status and future plans [A]. Linux Symposium [C]. 2007：21－33.

[5] 王兆威. 基于块设备驱动的安卓系统存储保护技术研究 [D]. 北京：北京理工大学，2015.

[6] Park D，Shin D. Removing duplicated writes at DB checkpointing with file system－level block remapping [A]. ACM International Conference on Computing Frontiers [C]. ACM，2015：37.

[7] 刘卫东. 移动终端闪存文件系统的性能分析与优化技术研究 [D]. 长沙：湖南大学，2014.

[8] Lee C，Sim D，Hwang J Y，et al. F2FS：a new file system for flash storage [A]. Usenix Conference on File and Storage Technologies [C]. USENIX Association，2015：273－286.

[9] 陈 颖. Linux 日志型文件系统的研究及其性能优化 [D]. 中国科学技术大学，2009.

[10] Surhone L M，Timpledon M T，Marseken S F，et al. Journaling Block Device [M]. Betascript Publishing，2010.

[11] 王建勋. 基于 NAND 闪存的固态存储技术研究与实现 [D]. 国防科学技术大学，2010.

[12] 赵 培. 闪存的存储管理及索引方法研究 [D]. 武汉：华中科技大学，2011.

[13] 朱康挺. 一种 NAND Flash 的垃圾回收及块管理方法的设计 [D]. 南京：东南大学，2012.

(上接第 269 页)

[7] Glorot X，Bordes A，Bengio Y. Deep sparse rectifier neural networks [A]. International Conference on Artificial Intelligence and Statistics [C]. 2012：315－323.

[8] Gkioxari G，Malik J. Finding action tubes [Z]. 2014：759－768.

[9] Wagner R，et al. Learning convolutional neural networks from few samples [A]. International Joint Conference on Neural Networks [C]. IEEE，2013：1－7.

[10] Oquab M，Bottou L，Laptev I，et al. Learning and Transferring Mid－level Image Representations Using Convolutional Neural Networks [A]. Computer Vision and Pattern Recognition [C]. IEEE，2014：1717－1724.

[11] Brox T，Bruhn A，Papenberg N，et al. High accuracy optical flow estimation based on a Theory for warping [Z]. 2004，3024 (10)：25－36.