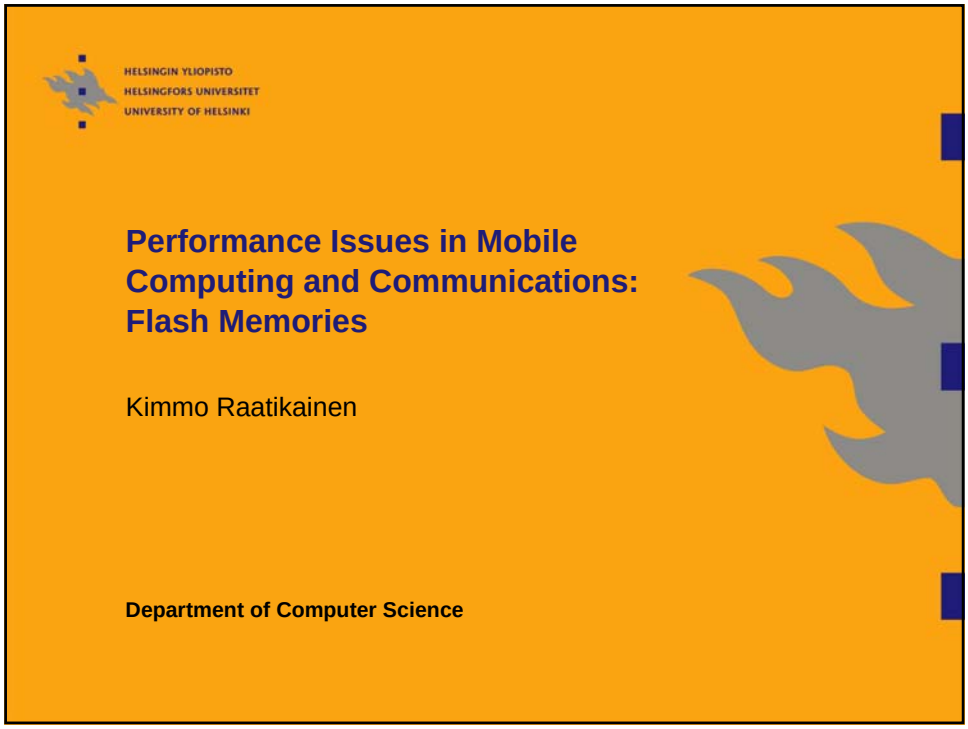
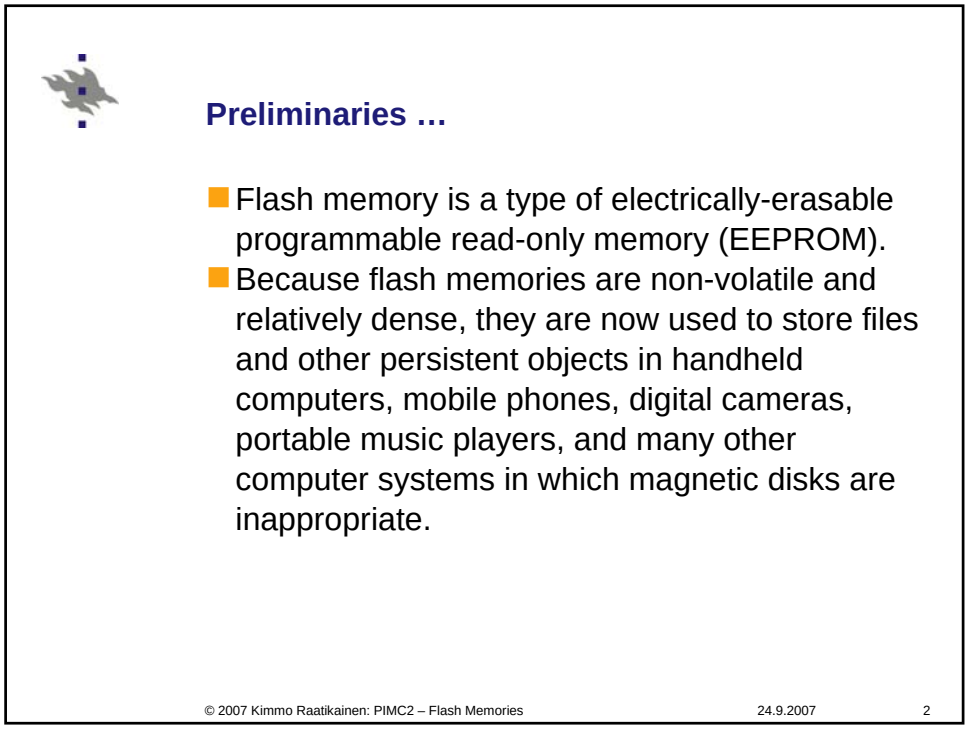

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI
Performance Issues in Mobile Computing and Communications: Flash Memories
Kimmo Raatikainen
Department of Computer Science


Preliminaries …
Flash memory is a type of electrically-erasable programmable read-only memory (EEPROM).
Because flash memories are non-volatile and relatively dense, they are now used to store files and other persistent objects in handheld computers, mobile phones, digital cameras, portable music players, and many other computer systems in which magnetic disks are inappropriate.
© 2007 Kimmo Raatikainen: PIMC2 – Flash Memories
24.9.2007
2

… Preliminaries

- Flash, like earlier EEPROM devices, suffers from two limitations.
  - First, bits can only be cleared by erasing a large block of memory.
  - Second, each block can only sustain a limited number of erasures, after which it can no longer reliably store data.
- Due to these limitations, sophisticated data structures and algorithms are required to effectively use flash memories.
- These algorithms and data structures support efficient not-in-place updates of data, reduce the number of erasures, and level the wear of the blocks in the device.

Lecture Outline

- Basic Facts
- Block-Mapping Techniques
- Flash-Specific File Systems
- Additional Classes of Flash Data Structures
  - application-specific data structures (mainly search trees),
  - data structures for storing machine code, and
  - a mechanism to use flash as a main memory replacement.

- Eran Gal and Sivan Toledo: ”Algorithms and Data Structures for Flash Memories,” ACM Computing Surveys, Vol. 37, No. 2, June 2005, pp. 138-163.

Basic Facts …

- The read/write/erase behaviours of flash memory is radically different than that of other programmable memories such as volatile RAM and magnetic disks.
- Perhaps more importantly, memory cells in a flash device (as well as in other types of EEPROM memory) can be written to only a limited number of times, between 10,000 and 1,000,000, after which they wear out and become unreliable.

… Basic Facts …

- Flash memories come in two flavours NOR and NAND.
- They are quite different.
- In both types, write operations can only clear bits (change their value from 1 to 0).
- The only way to set bits (change their value from 0 to 1) is to erase an entire region memory.
- These regions have fixed size in a given device, typically ranging from several kilobytes to hundreds of kilobytes and are called erase units.

… Basic Facts …

- NOR flash (the older type) is a random-access device that is directly addressable by the processor.
  - Each bit in a NOR flash can be individually cleared once per-erase-cycle of the erase unit containing it.
  - NOR devices suffers from high erase times.
- NAND flash (the newer type) enjoys much faster erase times, but it is not directly addressable
  - it is accessed by issuing commands to a controller,
  - access is by page (a fraction of an erase unit, typically 512 bytes) not by bit or byte, and
  - each page can be modified only a small number of times in each erase cycle.

… Basic Facts

- Because of these peculiarities, storage management techniques that were designed for other types of memory devices are not always appropriate for flash.
- To address these issues, flash-specific storage techniques have been developed with the widespread introduction of flash memories in the early 1990s.
- Some of these techniques were invented specifically for flash memories, but many have been adapted from techniques that were originally invented for other storage devices.

… Basic Facts …

- Because of these peculiarities, storage management techniques that were designed for other types of memory devices, such as magnetic disks, are not always appropriate for flash. To address these issues, flash-specific storage techniques have been developed with the widespread introduction of flash memories in the early 1990s. Some of these techniques were invented specifically for flash memories, but many have been adapted from techniques that were originally invented for other storage devices.
- The techniques that are used in some flash-management products have remained trade secrets; some are alluded to in corporate literature but are not fully described.

Block-Mapping Techniques

Block-Mapping Techniques …

- One approach to using flash memory is to treat it as a block device that allows fixed-size data blocks to be read and written much like disk sectors.
- This allows standard file systems designed for magnetic disks, such as FAT, to utilize flash devices.
- In this setup, the file system code calls a device driver, requesting block read or write operations.
- The device driver stores and retrieves blocks from the flash device.
  - Some removable flash devices, like CompactFlash, even incorporate a complete ATA disk interface so they can actually be used through the standard disk driver.

… Block-Mapping Techniques …

- Mapping the blocks onto flash addresses in a simple linear fashion presents two problems.
- First, some data blocks may be written to much more often than others.
  - When the file system in mapped onto a flash device, frequently used erase units wear out quickly, slowing down access times and eventually burning out.
  - This problem can be addressed by using a more sophisticated block-to-flash mapping scheme and by moving blocks around.
  - Techniques that implement such strategies are called wear-leveling techniques.

… Block-Mapping Techniques

- The second problem that the identity mapping poses is the inability to write data blocks smaller than a flash erase unit.
  - Suppose that the data blocks that the file system uses are 4KB each and that flash erase units are 128KB each.
  - If 4KB blocks are mapped to flash addresses using the identity mapping, writing a 4KB block requires copying a 128KB flash erase unit to RAM, overwriting the appropriate 4KB region, erasing the flash erase unit, and rewriting it from RAM.

The Block-Mapping Idea …

- The basic idea behind all the wear-leveling techniques is to map the block number presented by the host (virtual block number) to a physical flash address (sector).
- When a virtual block needs to be rewritten, the new data does not overwrite the sector where the block is currently stored.
  - The new data is written to another sector and the virtual-block-to-sector map is updated.
- Typically, sectors have a fixed size and occupy a fraction of an erase unit.
  - In NAND devices, sectors usually occupy one flash page.
  - But in NOR devices, it is also possible to use variable-length sectors.

… The Block-Mapping Idea …

- This mapping serves several purposes:
  - First, writing frequently modified blocks to a different sector in every modification evens out the wear of different erase units.
  - Second, the mapping allows writing a single block to flash without erasing and rewriting an entire erase unit.
  - Third, the mapping allows block writes to be implemented atomically so that, if power is lost during a write operation, the block reverts to its prewrite state when flash is used again.

… The Block-Mapping Idea …

- Atomicity is achieved using the following technique:
  - **Each sector is associated with a small header which may be adjacent to the sector or elsewhere in the erase unit.**
  - **When a block is to be written, the software searches for a free and erased sector.**
  - **In this state, all the bits in both the sector and its header are 1.**
  - **Then a free/used bit in the header of the sector is cleared, to mark that the sector is no longer free.**
  - **Then the virtual block number is written to its header, and the new data is written to the chosen sector.**
  - **Next, the prevalid/valid bit in the header is cleared to mark the sector is ready for reading.**
  - **Finally, the valid/obsolete bit in the header of the old sector is cleared to mark that it is no longer contains the most recent copy of the virtual block.**

… The Block-Mapping Idea

- If power is lost during a write operation, the flash may be in two possible states with respect to the modified block.
- If power was lost before the new sector was marked valid, its contents are ignored when the flash is next used, and its valid/obsolete bit can be set to mark it ready for erasure.
- If power was lost after the new sector was marked valid but before the old one was marked obsolete, both copies are legitimate, and the system can choose either one and mark the other obsolete.
- If choosing the most recent version is important, a two-bit version number can indicate which one is more recent.

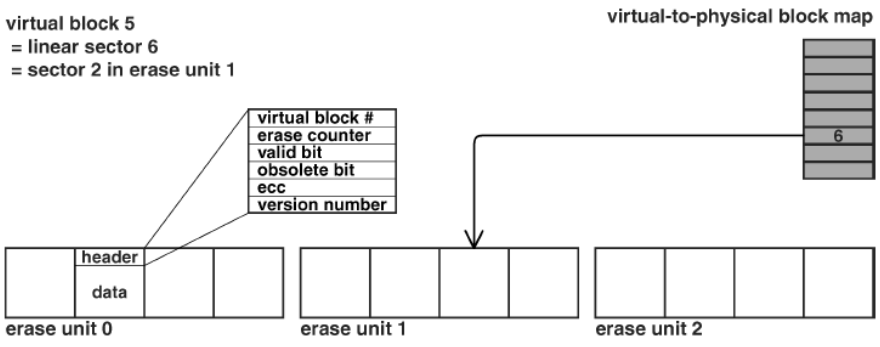
Data Structures for Mapping …

- Direct maps are essentially arrays that store in the i^{th} location the index of the sector that currently contains block i.
- Inverse maps store in the i^{th} location the identity of the block stored in the i^{th} sector.
- Inverse maps are stored on the flash device itself.
- When a block is written to a sector, the identity of the block is also written.
- The block's identity is always written in the same erase unit as the block itself so that they are erased together.
- The main use of the inverse map is to reconstruct a direct map during device initialization

… Data Structures for Mapping …

- The reason that direct maps are stored in RAM is that, by definition, they support fast lookups.
- This implies that when a block is rewritten and moved from one sector to another, a fixed lookup location must be updated.

Block mapping in a flash device

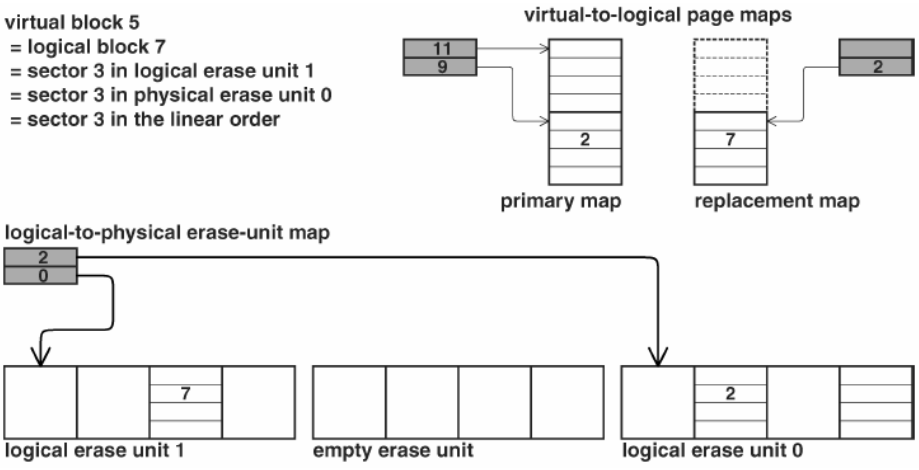


© ACM Computing Surveys, 37, 3 (June 2005) p. 142

Flash Translation Layer …

- The Flash Translation Layer (FTL) is a technique to store some of the direct map within the flash device itself while trying to reduce the cost of updating the map on the flash device.
- This technique was later adopted as a PCMCIA standard.

An example of the FTL mapping structures



© ACM Computing Surveys, 37, 3 (June 2005) p. 143

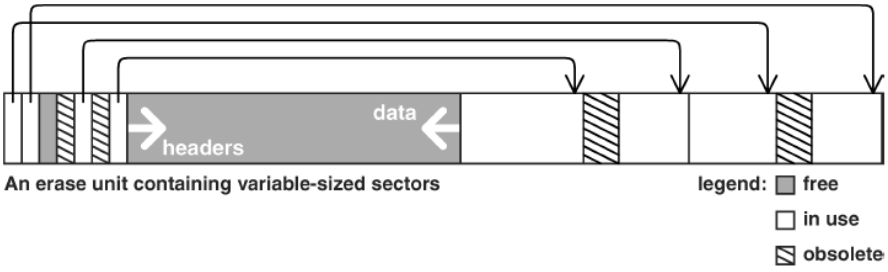
… Flash Translation Layer …

1. Block numbers are first mapped to logical block numbers which consist of a logical erase unit number and a sector index within the erase unit.
   - Each sector is copied to the same location in the new erase unit.
2. The mapping of the first blocks, which in FAT-formatted devices change frequently, can be stored in RAM, while the rest is stored in the flash device.
3. The flash portion of the block-to-logical-block map is not stored contiguously in the flash but is scattered throughout the device along with an inverse map.
   - The map is stored in a two-level hierarchical structure.

… Flash Translation Layer …

4. When a block is rewritten and moved to a new sector, its mapping must be changed.
   - This mechanism favours sequential modification of blocks since, in such cases, multiple mappings are moved from the main map to the backup map before a new mapping sector must be written.
   - The backup map can be sparse; not every mapping sector must have a backup sector.
5. Finally, logical erase units are mapped to physical erase units using a small direct map in RAM.
   - One entry per erase unit, not per sector.

Block mapping with variable-length sectors



© ACM Computing Surveys, 37, 3 (June 2005) p. 144

Erase-Unit Reclamation …

- Over time, the flash device accumulates obsolete sectors and the number of free sectors decrease.
- To make space for new blocks and for updated blocks, obsolete sectors must be reclaimed.
- Since the only way to reclaim a sector is to erase an entire unit, reclamation (garbage collection) operates on entire erase units.
- Reclamation can take place either in the background or on-demand when the amount of free space drops below a predetermined threshold.

… Erase-Unit Reclamation …

- The system reclaims space in several stages.
  - One or more erase units are selected for reclamation.
  - The valid sectors of these units are copied to newly allocated free space elsewhere in the device.
    - Copying the valid data prior to erasing the reclaimed units ensures persistence even if a fault occurs during reclamation.
  - The data structures that map logical blocks to sectors are updated, if necessary, to reflect the relocation.
  - Finally, the reclaimed erase units are erased and their sectors are added to the free-sector reserve.
    - This stage might also include writing an erase-unit header on each newly-erased unit.

… Erase-Unit Reclamation …

- In many cases, the system keeps at least one or two free and erased units at all times to allow all the valid data in a unit that is being reclaimed to be relocated to a single erase unit.
- The reclamation mechanism is governed by two policies:
  - which units to reclaim, and
  - where to relocate valid sectors to.
- These policies are related to another policy which governs sector allocation during block updates.

… Erase-Unit Reclamation …

- These three interrelated policies affect the system in three ways.
- They affect the effectiveness of the reclamation process which is measured by the number of obsolete sectors in reclaimed units
- They affect wear leveling
- They affect the mapping data structures (some relocations require simple map updates and some require complex updates).
- The goals of wear leveling and efficient reclamation are often contradictory.

… Erase-Unit Reclamation

- The fourth policy triggers reclamation events.
- Clearly, reclamation must take place when the system needs to update a block but no free sector is available.

Wear-Centric Reclamation Policies and Wear-Measuring Techniques

- Typically, the system uses an efficiency-centric reclamation policy most of the time but switches to a wear-centric technique that ignores efficiency once in a while.
- Sometimes uneven wear triggers the switch, and sometimes it happens periodically whether or not wear is even.
- Many techniques attempt to level the wear by measuring it.
- Most techniques are patented; for details see Gal&Toledo

Combining Wear-Leveling with Efficient Reclamation …

- One policy that attempt to address both wear leveling and efficient reclamation is based on a block device driver for a flash device.
  - The driver was intended for use with a log-structured Unix file system. It has two reclamation policies.
- The first policy selects the next unit for reclamation based on a weighted benefit/cost ratio.
  - The benefit of a unit reclamation is the amount of invalid space in the unit, and the cost is incurred by the need to read the valid data and write it back elsewhere.
  - This policy still leads to inefficient reclamation in some cases.

… Combining Wear-Leveling with Efficient Reclamation …

- The second policy tends to improve efficiency at the expense of worse wear leveling.
  - Data is written to two units.
  - One unit is used for sectors relocated during the reclamation of so-called cold units, ones that were not modified recently.
  - The other unit is used for sectors relocated from hot units and for updating blocks not undergoing reclamation. This policy tends to cluster static data in some units and dynamic data in others.

… Combining Wear-Leveling with Efficient Reclamation …

- This, in turn, tends to increase the efficiency of reclamation, because units with dynamic data tend to be almost empty upon reclamation, and static units do not need to be reclaimed at all because they often contain no invalid data.
- Clearly, units with static data can remain unreclaimed for long periods which leads to uneven wear unless a separate explicit wear-leveling mechanism is used.

… Combining Wear-Leveling with Efficient Reclamation …

- The eNVy system [Wu and Zwaenepoel 1994] partitions the erase units into fixed-size partitions.
- Lower-numbered partitions are supposed to store hot virtual blocks, while higher-numbered partitions are supposed to store cold virtual blocks.
- When a virtual block is updated, the new data is written to a sector in the active unit in the same partition that the block currently resides in.

… Combining Wear-Leveling with Efficient Reclamation …

- The system tries to achieve a roughly constant reclamation frequency in all the partitions.
- Therefore, hot partitions should contain fewer blocks than cold partitions because hot data tends to be invalidated more quickly.
- If the partition's reclamation frequency is higher than average, some of its blocks are moved to neighboring partitions.
- Blocks from the beginning of the unit being reclaimed are moved to a colder partition, and blocks from the end are moved to a hotter partition.

… Combining Wear-Leveling with Efficient Reclamation …

- Block clustering mechanisms called CAT and DAC were proposed in 1999.
- At the heart of these methods lies a concept called temperature.
- The temperature of a block is an estimate of the likelihood that it will be updated soon.
- The system maintains a temperature estimate for every block using two simple rules:
  - (1) when a block is updated, its temperature rises, and
  - (2), blocks cool down over time.
- The CAT policy classifies blocks into three categories: read-only (absolutely static), cold, and hot.

… Combining Wear-Leveling with Efficient Reclamation

- The DAC policy is more sophisticated.
- First, blocks may be classified into more than three categories.
- More importantly, blocks are reclassified on every update so a cold block that heats up will be relocated to a hotter erase unit, even if the units that store it never get reclaimed while it is valid.

Real-Time Reclamation

- Chang et al. [ACM TOECS 3, 4 (2004) 837–863.] proposed a guaranteed reclamation policy for real-time systems with periodic tasks.
- They assume that tasks are periodic, and that each task provides the system with its period, with per-period upper bounds on CPU time and the number of sector updates.
- The system uses a greedy reclamation policy that reclaims the unit with the least amount of valid data, and it only reclaims units when the number of free sectors falls below a threshold.
  - This policy is used only under deadline pressure; for non-realtime reclamation, a different policy is used. It ensures that every reclamation generates a certain number of free sectors.

Flash-Specific File Systems

Flash-Specific File Systems

- Another approach is to expose the hardware characteristics of the flash device to the file-system layer and let it manage erase units and wear.
- The article has a good background section (3.1) on Log-Structured File Systems

The Research-In-Motion File System …

- Research In Motion (http://www.rim.net) , a company making handheld text messaging devices and smart phones, patented a log-structured file system for flash memories.
  - The file system is designed to store contiguous variable-length records of data, each having a unique identifier.
- The flash is partitioned into an area for programs and an area for the file system
  - this is fairly common and is designed to allow programs to be executed in-place directly from NOR flash memory

… The Research-In-Motion File System

- The file system area is organized as a perfectly circular log containing a sequence of records.
  - Each record starts with a header containing the record's size, identity, invalidation
- Keeping the records contiguous allows the file system to return pointers directly into the NOR flash in read operation.

The Journaling Flash Filing System

- The Journaling Flash File System (JFFS) was originally developed by Axis Communications embedded Linux
  - [http://developer.axis.com/software/jffs/]
- It was later enhanced, in a version called JFFS2, by Red Hat [http://sources.redhat.com/jffs2/jffs2.pdf].
- Both versions are freely available under the GPL.
- Both versions focus mainly on NOR devices and may not work reliably on NAND devices.

… The Journaling Flash Filing System …

- JFFS2 is a posix compliant file system.
- Files are represented by an inode number.
- Inode numbers are never reused, and each version of an inode structure on flash carries a version number.
- Version numbers are also not reused.
- The version numbers allow the host to reconstruct a direct inode map from the inverse map stored on flash.

… The Journaling Flash Filing System

- At mount time, the system scans all the nodes in the log and builds two data structures.
- One is a direct map from each inode number to the most recent version of it on flash.
- This map is kept in a hash table.
  - The other is a collection of structures that represent each valid node on the flash.
  - Each structure participates in two linked lists
  - one chaining all the nodes according to physical address (to assist in garbage collection), and
  - the other containing all the nodes of a file, in order.

YAFFS: Yet Another Flash Filing System …

- YAFFS was written as a NAND file system for embedded device.
- It has been released under the GPL and has been used in products running both Linux and Windows CE.
- In YAFFS, files are stored in fixed-sized chunks which can be 512 bytes, 1KB, or 2KB in size.
- The file system relies on being able to associate a header with each chunk.
- The header is 16 bytes for 512 bytes chunks, 30 bytes for 1KB, and 42 bytes for 2KB.

… YAFFS: Yet Another Flash Filing System

- Each file (including directories) include one header chunk, containing the file's name and permissions, and zero or more data chunks.
- As in JFFS2, the only mapping information on flash is the content of each chunk, stored as part of the header.
- This implies that at mount time all the headers must be read from flash to construct the file ID and file contents maps, and that, as in JFFS, the maps of all the files are stored in RAM at all times.

The Trimble File System

- The Trimble file system was a NOR implemented for GPS equipment: Trimble Navigation.
- The overall structure of the file system is fairly similar to that of YAFFS.
- Files are broken into 252-byte chunks, and each chunk is stored with a 4-byte header in a 256-byte flash sector.
- The 4-byte header includes the file number and chunk number within the file.
- Each file also includes a header sector, containing the file number, a valid/invalid word, a file name, and up to 14 file records, only one of which, the last one, is valid.
- Each record contains the size of the file, a checksum, and the last modification time.

The Microsoft Flash File System

- In the mid 1990's, Microsoft tried to promote a standardized file system for removable flash memories which was called FFS2.
- Douglis et al. ["Storage alternatives for mobile computers." In Proceedings of the 1st USENIX Symposium on Operating Systems Design and Implementation (OSDI). Monterey, Calif., 1994, pp. 25–37.] report very poor write performance for this system which is probably the main reason it failed.
- By 1998, Intel [Flash file system selection guide. Application Note 686, Intel Corporation.] listed this solution as obsolete.

Norris Flash File System

- Norris Communications Corporation patented a flash file system based on linked lists, much like the Microsoft Flash File System.
- The file system was designed and implemented for use in a handheld audio recorder.

Other Commercial Embedded File Systems …

- TargetFFS.
  - Blunk Microsystems offers TargetFFT, an embedded flash file system, in both NAND and NOR varieties.
  - It under their own operating system but is designed to be portable to other operating systems and to products without an operating system.
  - The file system uses a POSIX-like API.
- smxFFS.
  - This file system from Micro Digital only supports nonremovable NAND devices.
  - The file system consists of a block-mapping device driver and a simple FAT-like file system with a flat directory structure.

… Other Commercial Embedded File Systems

- EFFS.
  - HCC Embedded offers several flash file systems.
  - EFFS-STD is a full file system.
  - EFFS-FAT and EFFS-THIN are FAT implementation for removable flash memories.
  - The two versions offer similar functionality, except that EFFS-THIN is optimized for 8-bit processors.
- FLite.
  - FLite combines a standard FAT file system with an FTL-compatible block-mapping device driver.
  - It is unclear whether this is a current product.

Beyond File Systems

Flash-Aware Application-Specific Data Structures

- Wu et al. proposed flash-aware implementations of B-trees and R-trees.
  - Their implementations represent a tree node as an ordered set of small items.
  - Items in a set represent individual insertions, deletions, and updates to a node.
  - The items are ordered by time so the system can construct the current state of a tree node by traversing its set of items.
- The main problem with flash-aware application-specific data structures is that they require that the flash device be partitioned.
  - One partition holds the application-specific data structure, another holds other data, usually files.

Execute-in-Place …

- Code stored in a NOR device, which is directly addressable by the processor, can be executed from the device itself without being copied into RAM.
  - This is known as execute-in-place, or XIP.
- Unless the system uses a virtual memory mechanism, which requires a hardware memory-management unit (MMU), code must be contiguous in flash, and it must not move.
- To implement XIP in a system without virtual memory requires partitioning the flash memory at the physical address level into a code partition and a data partition.

… Execute-in-Place

- Even if the system does use virtual memory, implementing XIP is tricky
- First, this requires that the block-mapping device driver be able to return the physical addresses that correspond to a given virtual block number.
- Second, the block-mapping device driver must notify the virtual memory subsystem whenever memory-mapped sectors are relocated.
- In addition, using XIP requires that code be stored uncompressed.

Flash-Based Main Memories …

- The eNVy is a flash-based nonvolatile main memory.
- The system was designed to reside on the memory bus of a computer and to service single-word read and write requests.
- The memory system itself contained a large NOR flash memory that was connected by a very wide bus to a battery-backed static RAM device.

… Flash-Based Main Memories

- The system partitions the physical address space of the external memory bus into 256-byte pages that are normally mapped by the internal MMU to flash pages.
- Read requests are serviced directly from this memory-mapped flash.
- Write requests are serviced by copying a page from the flash to the internal RAM, modifying the internal MMU's state so that the external physical page is now mapped to the RAM, and then performing the word-size write onto the RAM.

Summary

Summary …

- Flash memories have been an enabling technology for the introduction of computers into numerous hand held devices.
- A decade ago, flash memories were used mostly in boot loaders (BIOS chips) and as disk replacements for ruggedized computers.
  - Today, flash memories are also used in mobile phones and PDA's, portable music players and audio recorders, digital cameras, USB memory devices, remote controls, and more.
- Flash memories provide these devices with fast and reliable storage capabilities thanks to the sophisticated data structures and algorithms.

… Summary

- In general, the challenges posed by newer flash devices are greater than those posed by older devices—the devices are becoming harder to use.
  - This happens because flash hardware technology is driven mostly by the desire for increased capacity and performance often at the expense of ease of use.
  - This trend requires development of new software techniques and new system architectures for new types of devices.
- Unfortunately, many of these techniques are only described in patents, not in technical articles.