Configurability of Performance and Overheads in Flash Management

Tei-Wei Kuo, Jen-Wei Hsieh, Li-Pin Chang, and Yuan-Hao Chang

Department of Computer Science
& Information Engineering
Grad. Inst. Of Networking & Multimedia
National Taiwan University

Agenda

- Introduction
- Management Issues
- Performance vs Overheads
- Other Challenging Issues
- Conclusion

Introduction – Why Flash Memory

- Diversified Application Domains
  - Portable Storage Devices
  - Critical System Components
  - Consumer Electronics
  - Industrial Applications

Introduction – The Reality

- Tremendous Driving Forces from Application Sides
  - Excellent Performance
  - Huge Capacity
  - High Energy Efficiency
  - Reliability
  - Low Cost
  - Good Operability in Critical Conditions

Introduction – The Characteristics of Storage Media

Media	Access time		
	Read	Write	Erase
DRAM	60ns (2B) 2.56*us* (512B)	60ns (2B) 2.56*us* (512B)	-
NOR FLASH	150ns (1B) 14.4*us* (512B)	211*us* (1B) 3.52ms (512B)	1.2s (16KB)
NAND FLASH	10.2*us* (1B) 35.9*us* (512B)	201*us* (1B) 226*us* (512B)	2ms (16KB)
DISK	12.4ms (512B) (average)	12.4 ms(512B) (average)	-

[Reference] DRAM:2-2-2 PC100 SDRAM. NOR FLASH: Intel 28F128J3A-150. NAND FLASH: Samsung K9F5608U0M. Disk: Segate Barracuda ATA II.[1]

1. J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho. A space-efficient flash translation layer for compact-flash systems. *IEEE Transactions on Consumer Electronics*, 48(2):366–375, May 2002.

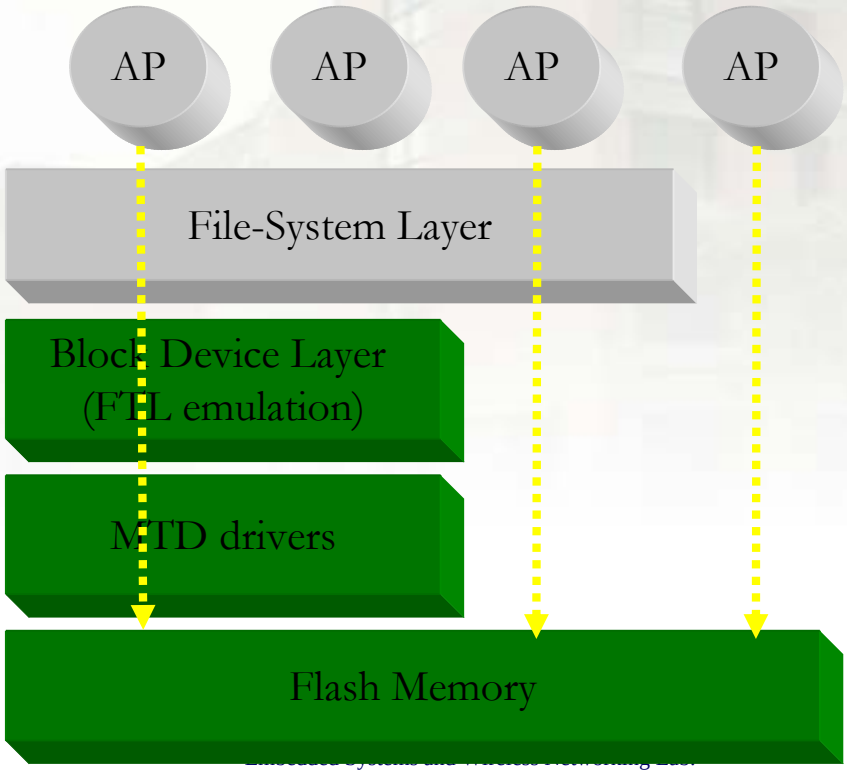
Introduction – Challenges

- Requirements in Good Performance
- Limited Cost per Unit
- Strong Demands in Reliability
- Increasing in Access Frequencies
- Tight Coupling with Other Components
- Low Compatibility among Vendors

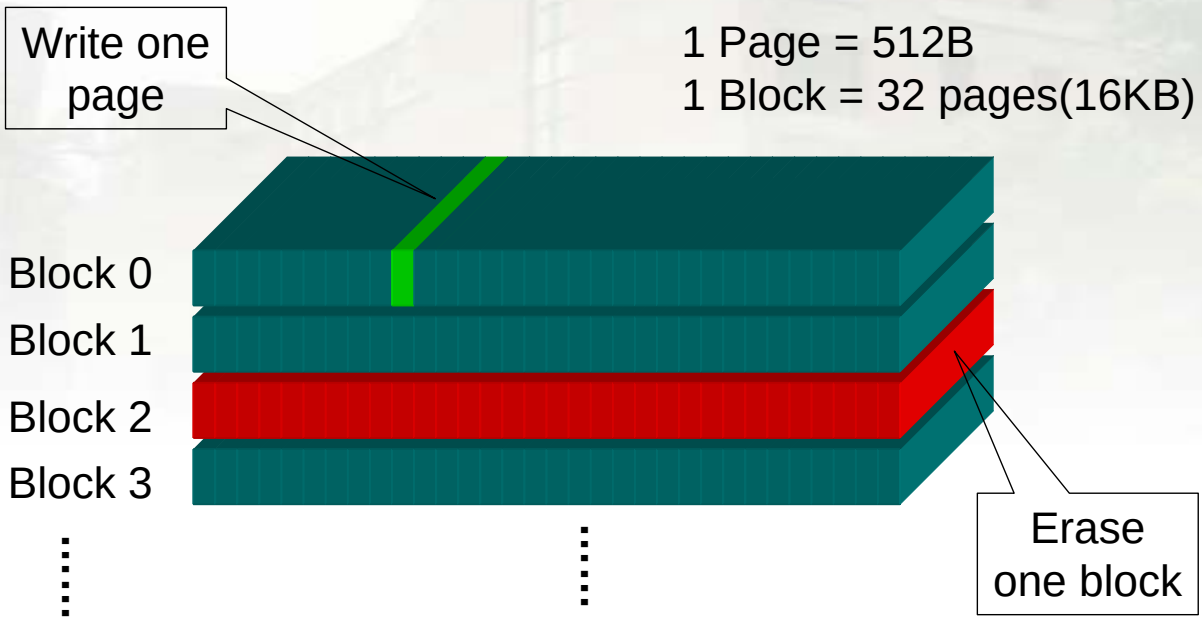
Agenda

- Introduction
- Management Issues
- Performance vs Overheads
- Other Challenging Issues
- Conclusion

Management Issues – System Architectures



Management Issues – Flash-Memory Characteristics

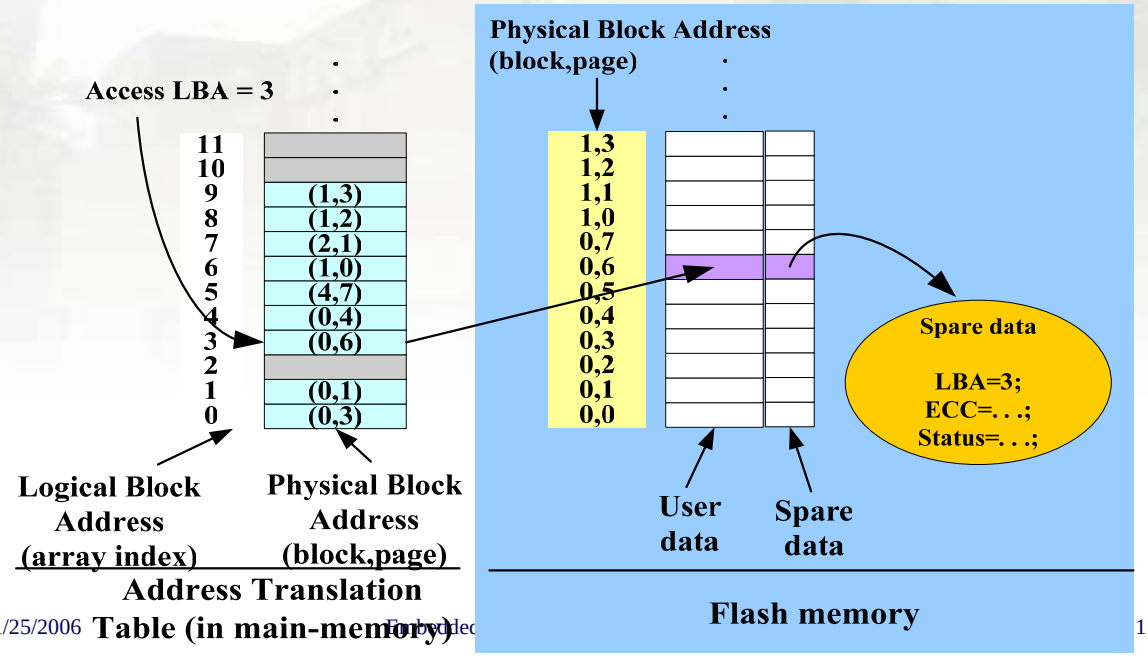


Management Issues – Flash-Memory Characteristics

- Write-Once
  - No writing on the same page unless its residing block is erased!
  - Pages are classified into valid, invalid, and free pages.
- Bulk-Erasing
  - Pages are erased in a block unit to recycle used but invalid pages.
- Wear-Leveling
  - Each block has a limited lifetime in erasing counts.

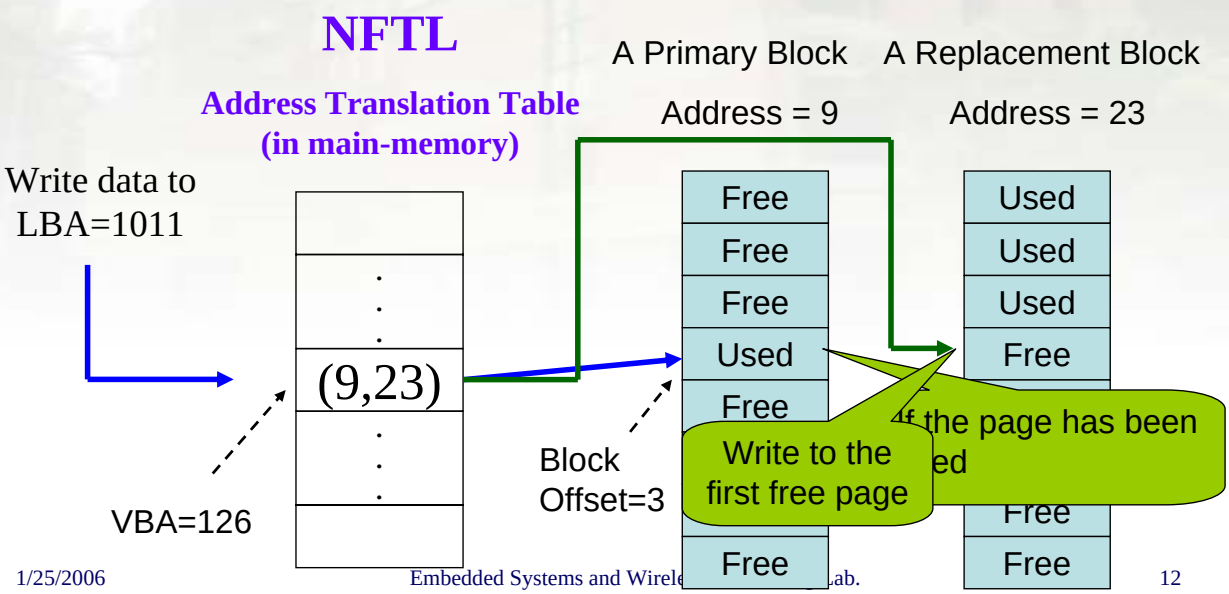
Management Issues – Policies: FTL

- FTL adopts a page-level address translation mechanism.
  - The main problem of FTL is on large memory space requirements for storing the address translation information.



Management Issues – Policies: NFTL

- A logical address under NFTL is divided into a virtual block address and a block offset.
  - e.g., LBA=1011 => virtual block address (VBA) = 1011 / 8 = 126 and block offset = 1011 % 8 = 3



Management Issues – Policies: NFTL

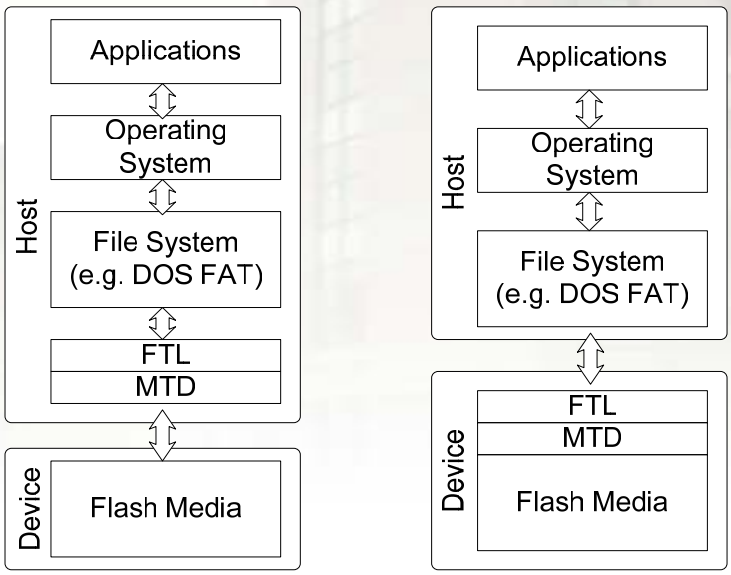
- NFTL is proposed for the large-scale NAND flash storage systems because NFTL adopts a block-level address translation.
- However, the address translation performance of read and write requests might deteriorate, due to linear searches of address translation information in primary and replacement blocks.

Management Issues – Policies

	FTL	NFTL
Memory Space Requirements	Large	Small
Address Translation Time	Short	Long
Garbage Collection Overhead	Less	More
Space Utilization	High	Low

- The Memory Space Requirements for one 256MB NAND (512B/Page, 4B/Table Entry, 32 Pages/Block)
  - FTL: 2,048KB (= 4*(256*1024*1024)/512)
  - NFTL: 64KB (= 4*(256*1024*1024)/(512*32))

Management Issues – Flash-Memory Characteristics



*FTL: Flash Translation Layer, MTD: Memory Technology Device

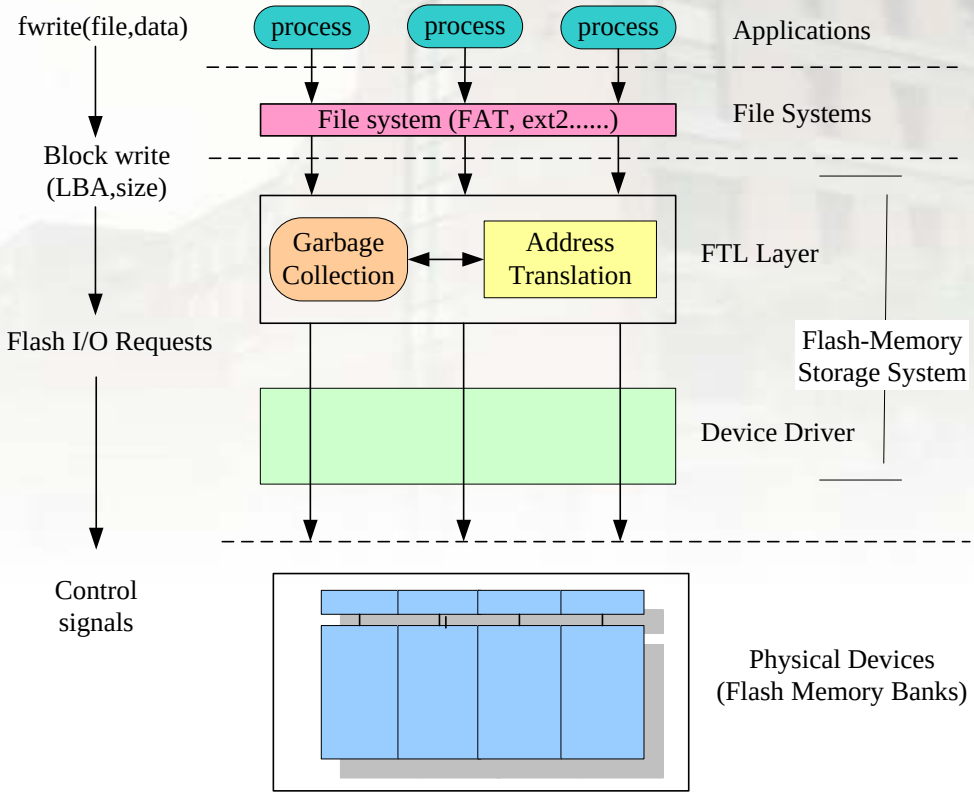
Management Issues – Observations

- The write throughput drops significantly after garbage collection starts!
- The capacity of flash-memory storage systems increases very quickly such that memory space requirements grows quickly.
- Reliability becomes more and more critical when the manufacturing capacity increases!
- The significant increment of flash-memory access numbers seriously exaggerates the Read/Program Disturb Problems!

Agenda

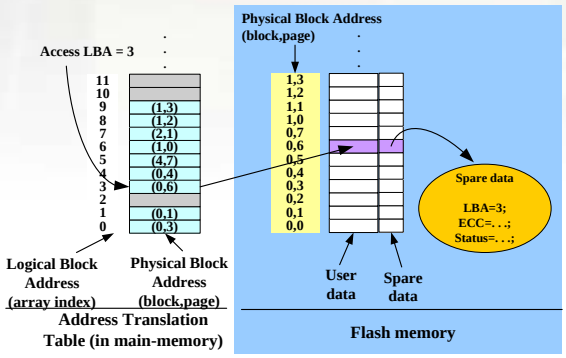
- Introduction
- Management Issues
- Performance vs Overheads
- Other Challenging Issues
- Conclusion

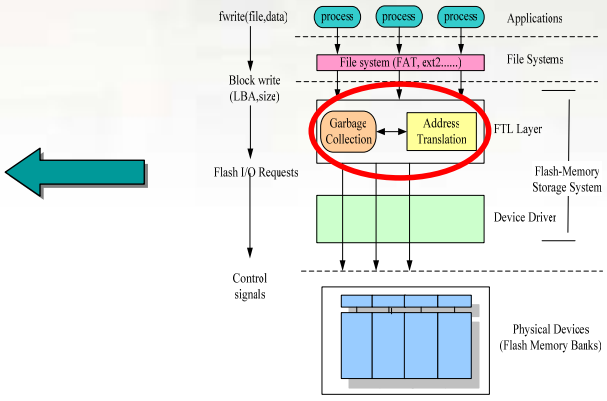
System Architecture



Flash Management

- Objectives
  - Performance
  - Memory Overheads
  - Space Utilization
  - Garbage Collection Cost





Flash Management

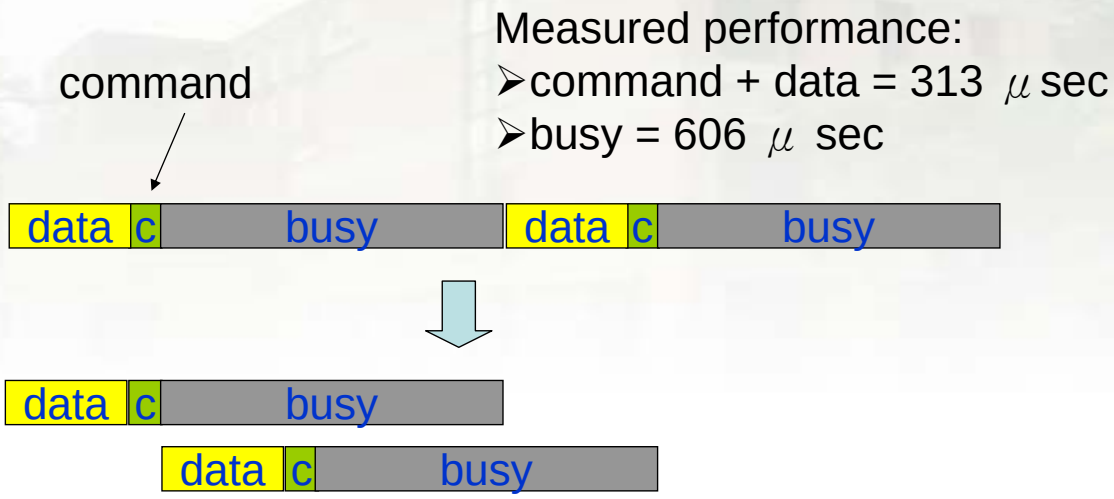
- Stripping Designs
- Efficient Hot-Data Identification
- Large-Scale Flash
- Reliability
- Address Translation Efficiency

Stripping Designs

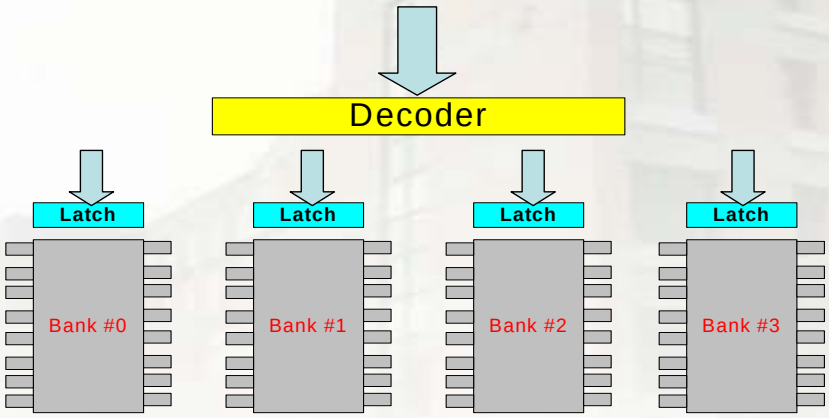
- Why?
  - Could we boost the system performance and enlarge the system capacity by simply having multiple flash banks working together?
- Issues
  - Space Utilization vs Wear-Leveling
  - Stripping Levels vs Performance
  - Performance vs Management Granularity

Stripping Designs – Parallelism

- An Example Parallelism in Write Operations



Stripping Designs



- Each bank can operate (read/write/erase) independently.
- One Common Technical Issue:
  - How to smartly distribute write requests among banks?

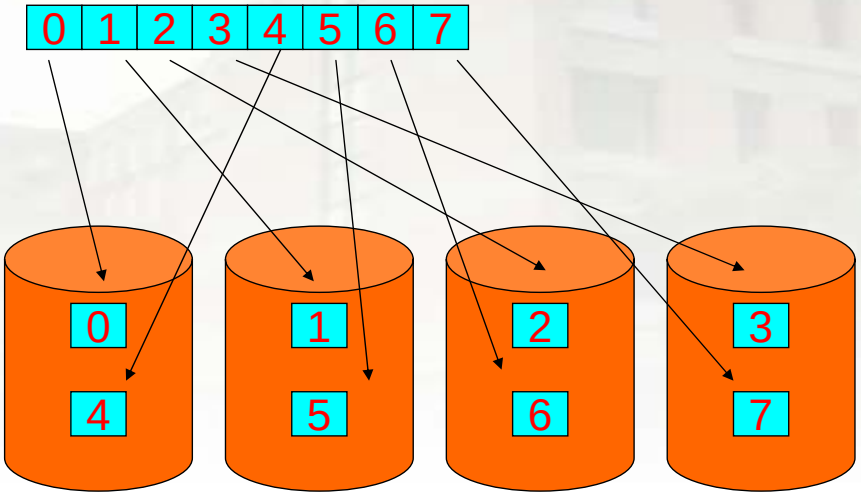
Stripping Designs

- Potential Issues
  - Static or Dynamic Stripping
    - Performance Boosting Bound?
  - Access Locality
    - Hot versus Cold Data

Stripping Designs – A Static Striping Policy

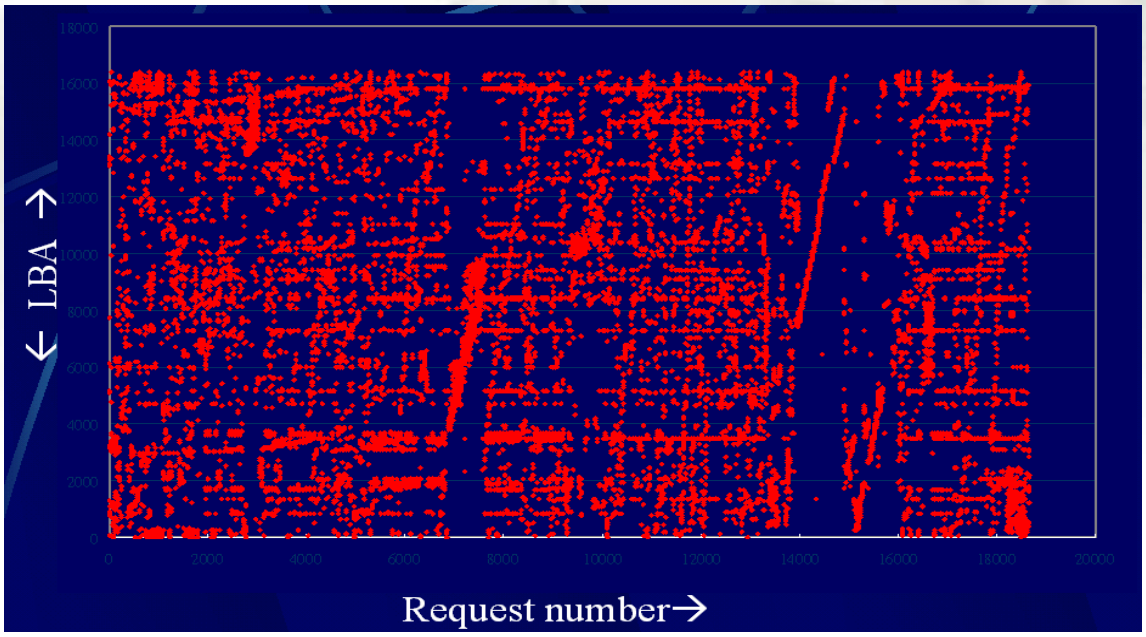
- A typical static striping policy would "evenly" scatter write requests over banks to improve the parallelism.
  - A RAID-0-Based Approach:
    - Bank address = (LBA) % (# of the total number of banks)

Stripping Designs – A Static Striping Policy



- True "fair usages" of banks could be hardly achieved by static striping!

Stripping Designs – A Snapshot of a Realistic Workload



Stripping Designs – Hot data

- Hot data usually come from
  - meta-data of file-systems, and
    - Small. A piece of hot data is usually $\leqq$ 2 sectors.
  - structured (or indexed) user files, etc.
- Storing of hot data on a statically assigned bank might
  - consume free space quickly,
  - start garbage collection frequently, or
  - wear their residing banks quickly.

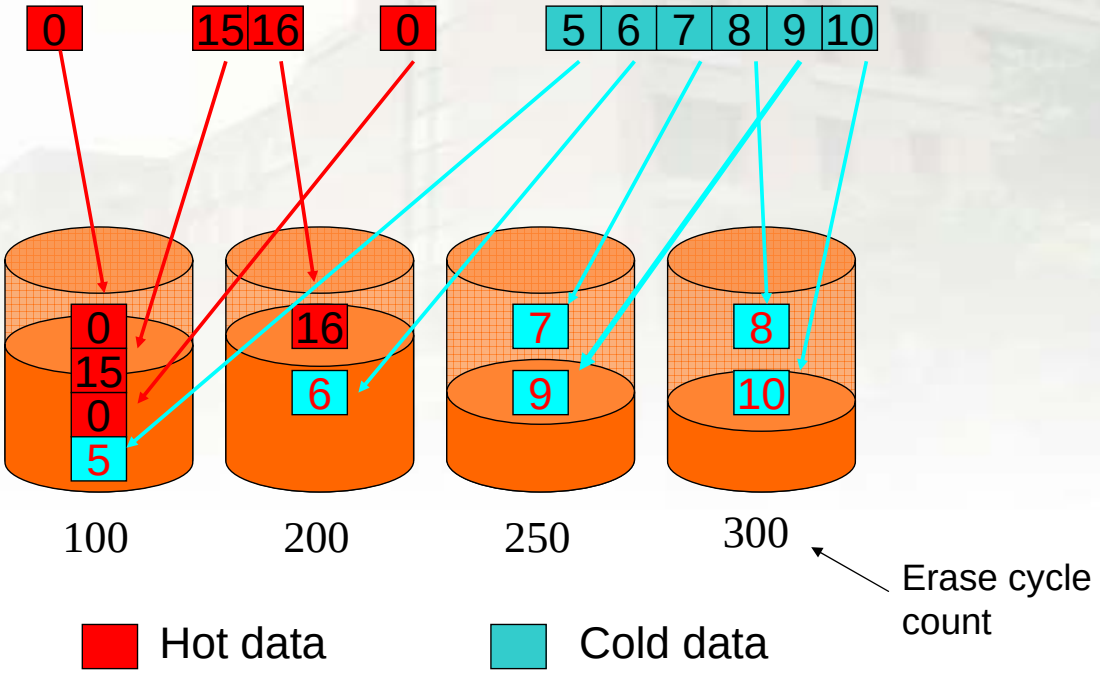
Stripping Designs – Cold Data

- Cold data usually come from
  - read-only (or WORM) files.
    - E.g., bulk and sequential files that often have a number of sectors.
- Storing of cold data on a statically assigned bank might
  - increase the capacity utilization, and
  - deteriorate the efficiency of garbage collection severely.

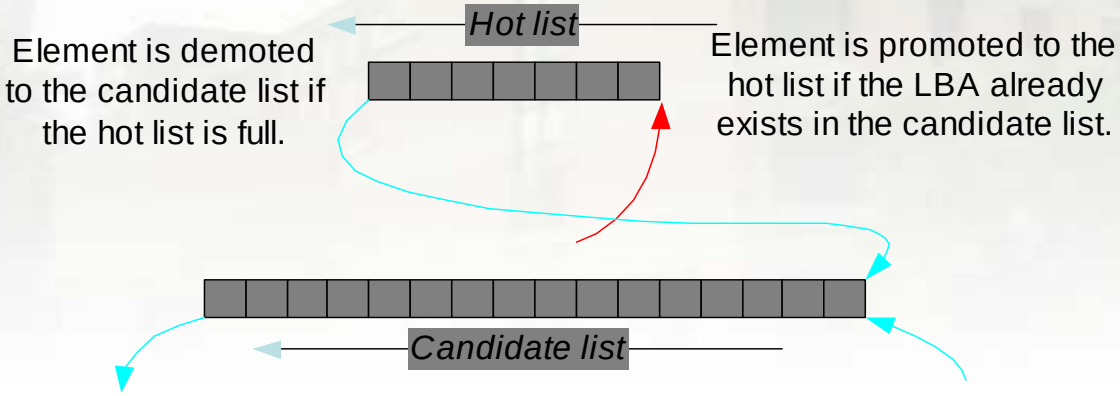
Stripping Designs – A Dynamic Striping Policy

- Main Strategies:
  - Distribute hot/cold data properly among banks.
    - Hot data → banks have low erase cycle counts.
    - Cold data → banks have low capacity utilizations.
- Remark: The hotness of written data should be efficiently identified!!!

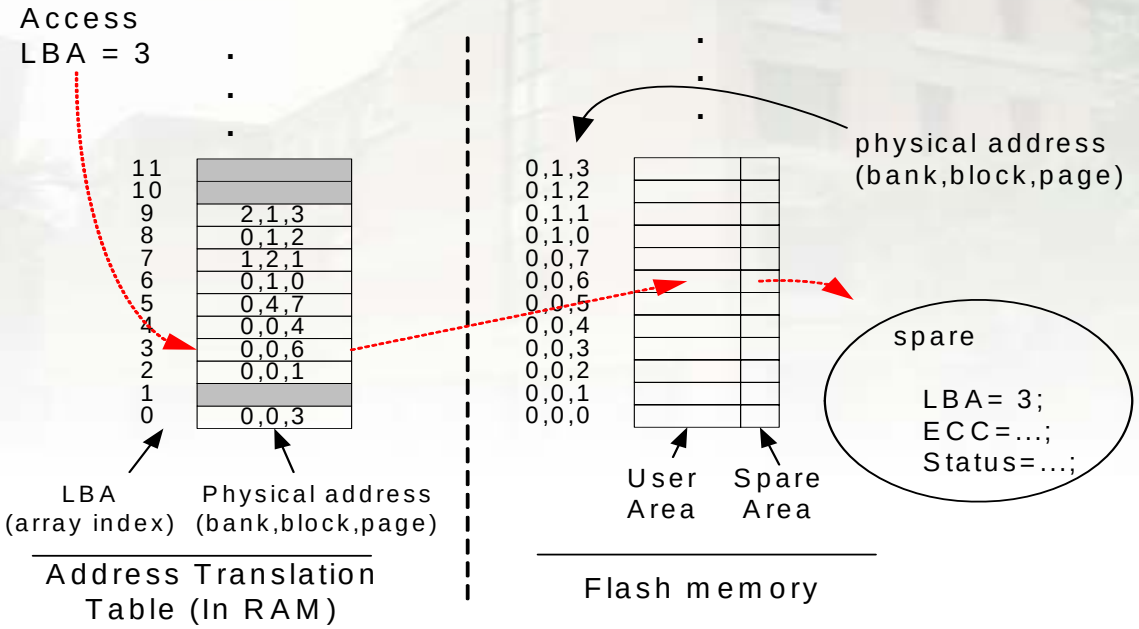
Stripping Designs – Dynamic Striping



Stripping Designs – A Hot-Cold Identification Mechanism

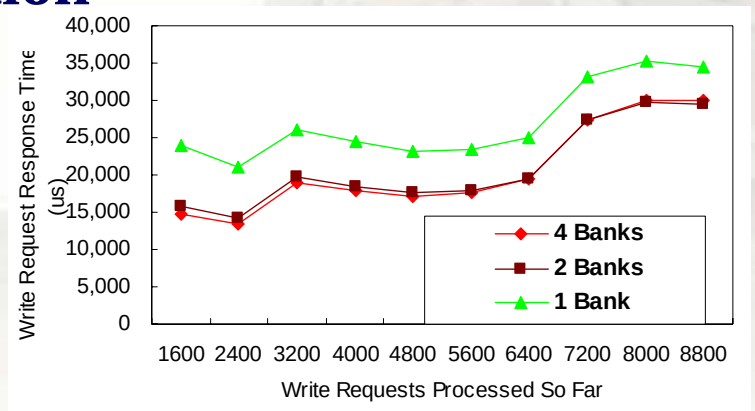


Stripping Designs – Address Translation



Stripping Designs – Performance Evaluation

When the Flash Capacity Is Fixed

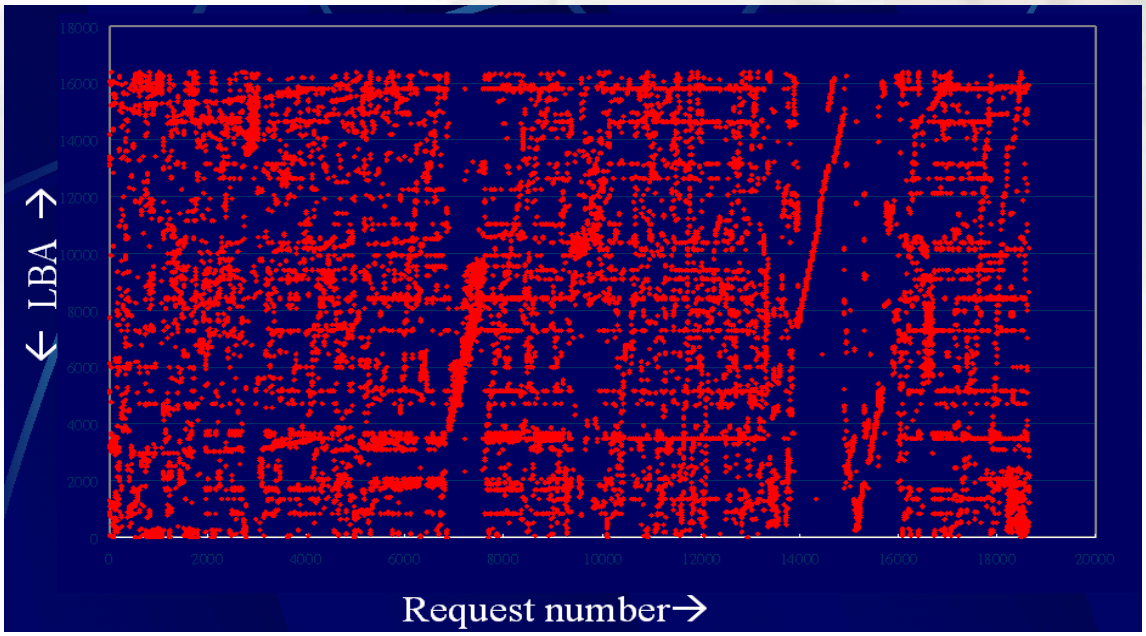


	Bank 0	Bank 1	Bank 2	Bank 3
Erase cycle counts (Dynamic)	350	352	348	350
Erase cycle counts (Static)	307	475	373	334
Capacity Utilization (Dynamic)	0.76	0.76	0.76	0.76
Capacity Utilization (Static)	0.72	0.81	0.77	0.74

Flash Management

- Stripping Designs
- Efficient Hot-Data Identification
- Large-Scale Flash
- Reliability
- Address Translation Efficiency

Efficient Hot-Data Identification – A Snapshot of a Realistic Workload



Efficient Hot-Data Identification – Why Important?

- Wear-Leveling
  - Pages that contain hot data could turn into dead pages very quickly.
  - Blocks with dead pages are usually chosen for erasing.

  Hot data should be written to blocks with smaller erase counts.
- Erase Efficiency (i.e., effective free pages reclaimed from garbage collection.)

  Mixture of hot data and non-hot data in blocks might deteriorate the efficiency of erase operations.

Efficient Hot-Data Identification

- Related Work
  - Maintain data update times for all LBA's (Logical Block Addresses)[1]

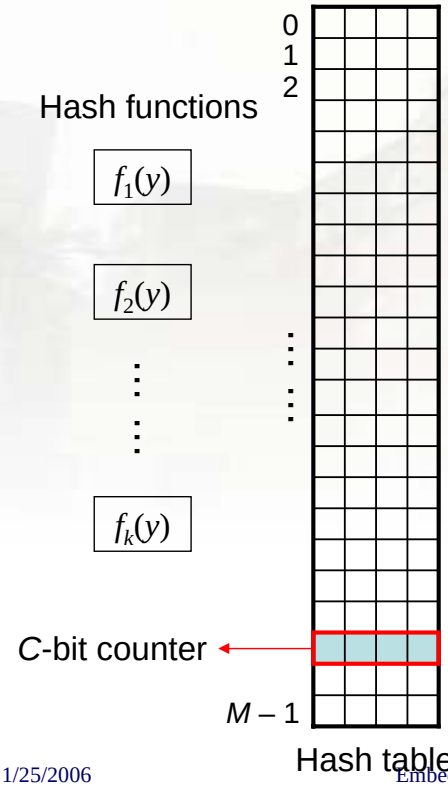
    Introduce significant memory-space overheads
  - Have a data structure to order LBA's in terms of their update times[2]

    Require considerable computing overheads
- Our Approach
  - A Multi-Hash-Function Framework
    - Identify hot data in a constant time
    - Reduce the required memory space

1. M. L. Chiang, Paul C. H. Lee, and R. C. Chang, "Managing Flash Memory in Personal Communication Devices," *ISCE* '97, December 1997, pp. 177-182

2. L. P. Chang and T. W. Kuo, "An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems," *8th IEEE RTAS*, September 2002, pp. 187-196

Efficient Hot-Data Identification – A Multi-Hash-Function Framework

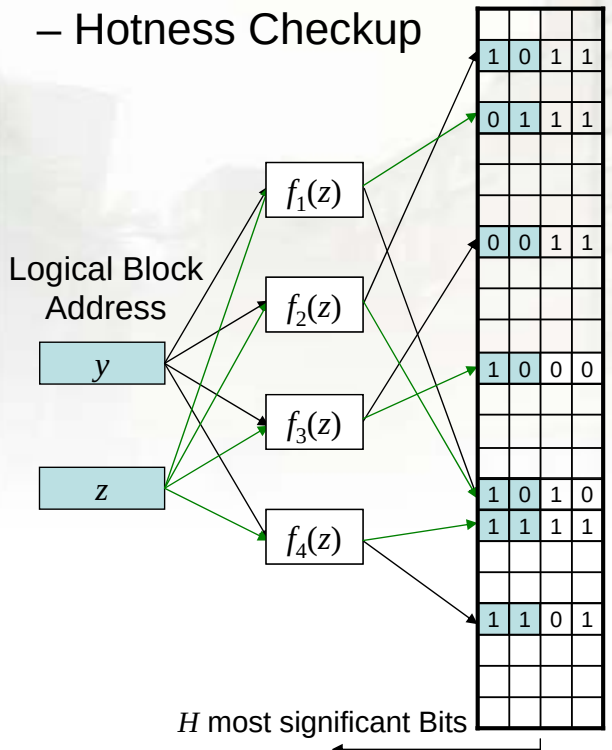


- *K* independent hash functions
- *M*-entry hash table
- *C*-bit counters

- Checkup
- Status Update
- Decay

Efficient Hot-Data Identification – A Multi-Hash-Function Framework

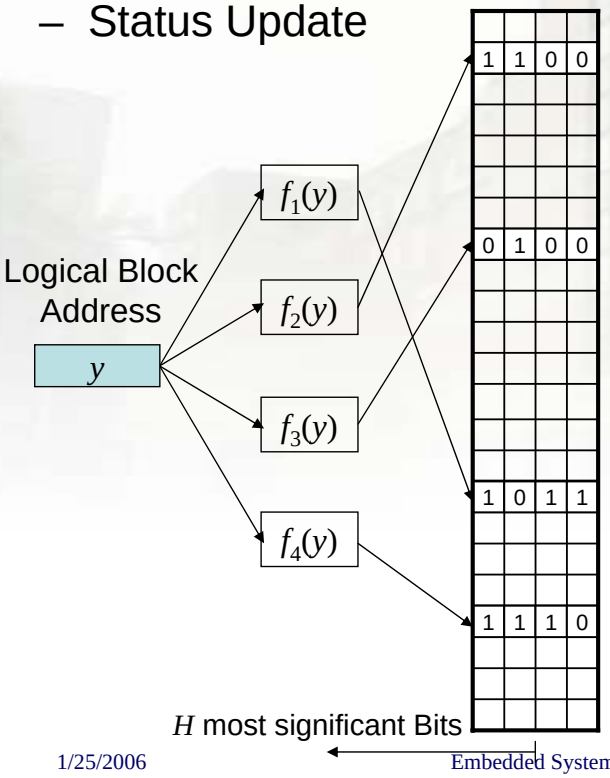
– Hotness Checkup



1. An LBA is to be verified as a location for hot data.
2. The corresponding LBA *y* is hashed simultaneously by *K* given hash functions.
3. Check if the *H* most significant bits of every counter of the *K* hashed values contain a non-zero bit value.

Efficient Hot-Data Identification – A Multi-Hash-Function Framework

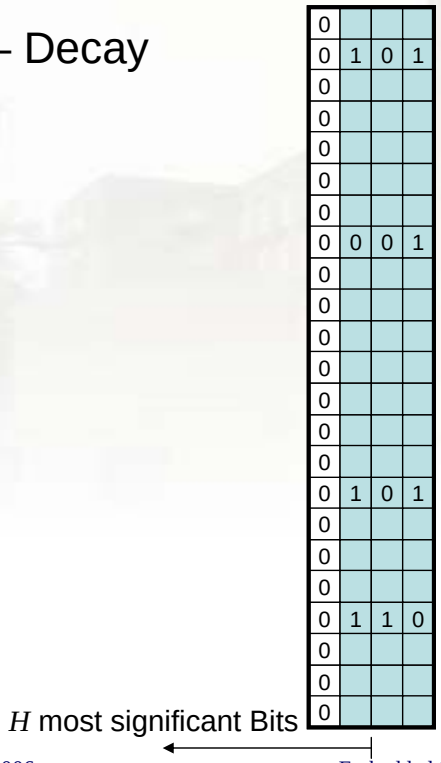
– Status Update



1. A write is issued to the FTL.
2. The corresponding LBA y is hashed simultaneously by K given hash functions.
3. Each counter corresponding to the K hashed values (in the hash table) is incremented by one to reflect the fact that the LBA is written again,

Efficient Hot-Data Identification – A Multi-Hash-Function Framework

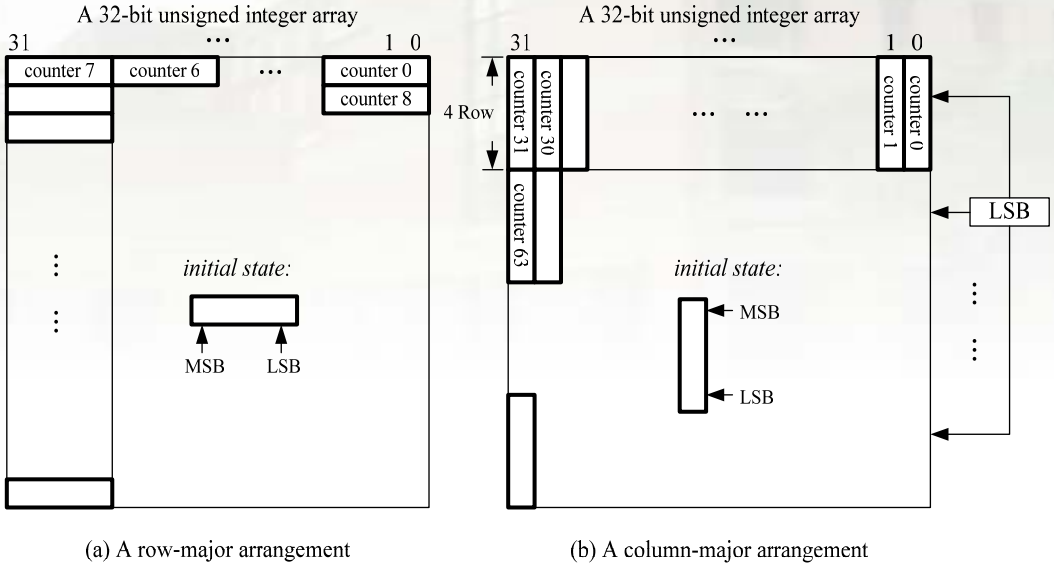
– Decay



For every given number of sectors have been written, called the “decay period” of the write numbers, the values of all counters are divided by 2 in terms of a right shifting of their bits.

Efficient Hot-Data Identification – Implementation Strategies

- A Column-Major Hash Table



(a) A row-major arrangement (b) A column-major arrangement

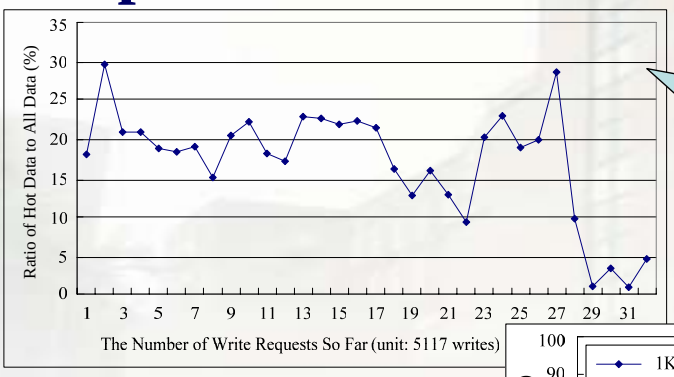
Efficient Hot-Data Identification – Analytic Study

- The probability of false identification of an LBA as a location for hot data:

$$(1 - (1 - 1/M)^{2NRK})^K - R$$

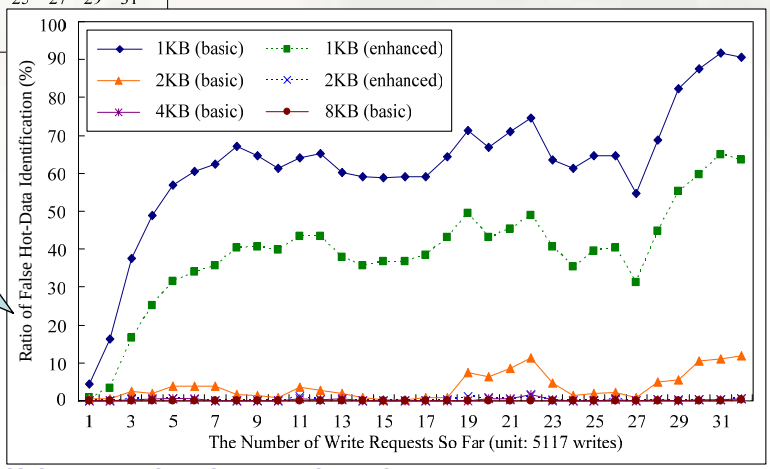
System Model Parameters	Notation
Number of Counters/Entries in a Hash Table	M
Number of Write References	N
Ratio of Hot Data in All Data (< 50%)	R
Number of Hash Functions	K

Efficient Hot-Data Identification – Impacts of Hash-Table Sizes



The locality of data access (decay period: 5117 writes)

Ratio of false hot-data identification for various hash-table sizes



Efficient Hot-Data Identification – Impacts of Decay Period

Efficient Hot-Data Identification – Runtime Overheads

	Multi-Hash-Function Framework (2KB)		Two-Level LRU List* (512/1024)	
	Average	Standard Deviation	Average	Standard Deviation
Checkup	2431.358	97.98981	4126.353	2328.367
Status Update	1537.848	45.09809	12301.75	11453.72
Decay	3565	90.7671	N/A	N/A

Unit: CPU cycles

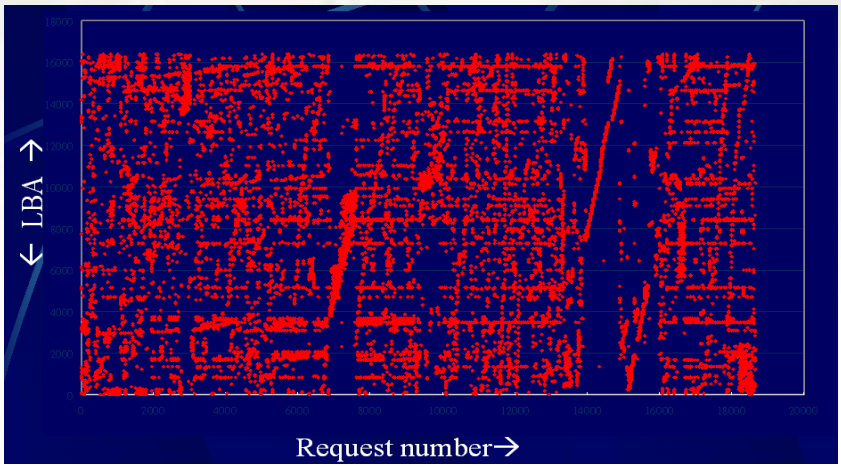
* L. P. Chang and T. W. Kuo, “An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems,” *8th IEEE RTAS*, September 2002, pp. 187-196

Flash Management

- Stripping Designs
- Efficient Hot-Data Identification
- Large-Scale Flash
- Reliability
- Address Translation Efficiency

Efficient Hot-Data Identification – A Snapshot of a Realistic Workload

- A Trace Collected for a Month over a 20GB Disk of a Personal Computer – Running Applications: Web browser, Email client, Multimedia applications, word processor, etc.

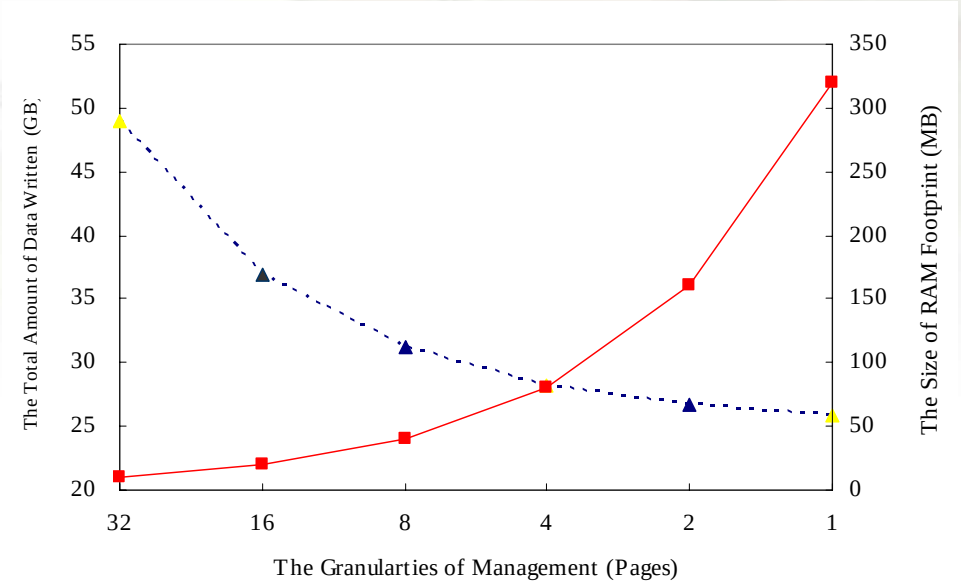


Large-Scale Flash – Observations

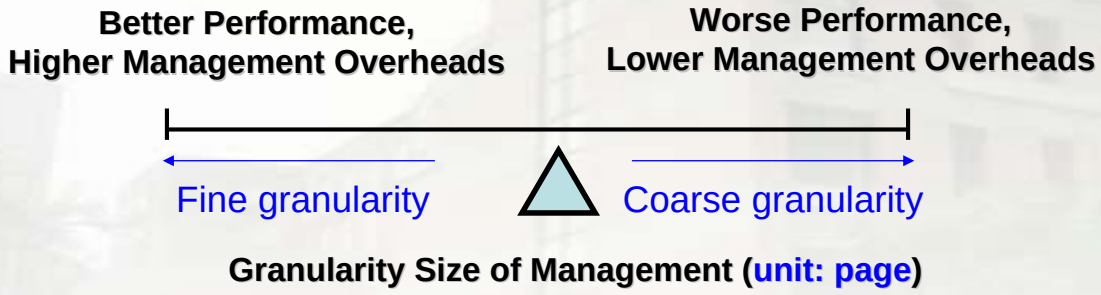
- Observations:
  - 62% of writes are no larger than 8 sectors.
    - Contribute 10% of total data written to the system.
    - Touch 1% of total LBA space.
    - Show a significant spatial locality.
  - 22% of writes are no less than 128 sectors.
    - Contribute 77% of total data written.
    - Touch 32% of total LBA space.
    - Tend to access the disk more sequentially.

Large-Scale Flash – Observations

- The trade-off between the main-memory usage and the system performance



Large-Scale Flash – Scalability



- **In a large-capacity flash memory, the management overheads and performance issues become more serious.**

- **Potential Tradeoffs**
  - Efficiency in Address Translation, Memory Space Requirements, Space Utilization, Garbage Collection Overheads!

Large-Scale Flash – Management Policy Revisiting!

	FTL	NFTL
Memory Space Requirements	Large	Small
Address Translation Time	Short	Long
Garbage Collection Overhead	Less	More
Space Utilization	High	Low

- The Memory Space Requirements for one 256MB NAND (512B/Page, 4B/Table Entry, 32 Pages/Block)
  - FTL: 2,048KB (= 4*(256*1024*1024)/512)
  - NFTL: 64KB (= 4*(256*1024*1024)/(512*32))

Large-Scale Flash – Address Translation

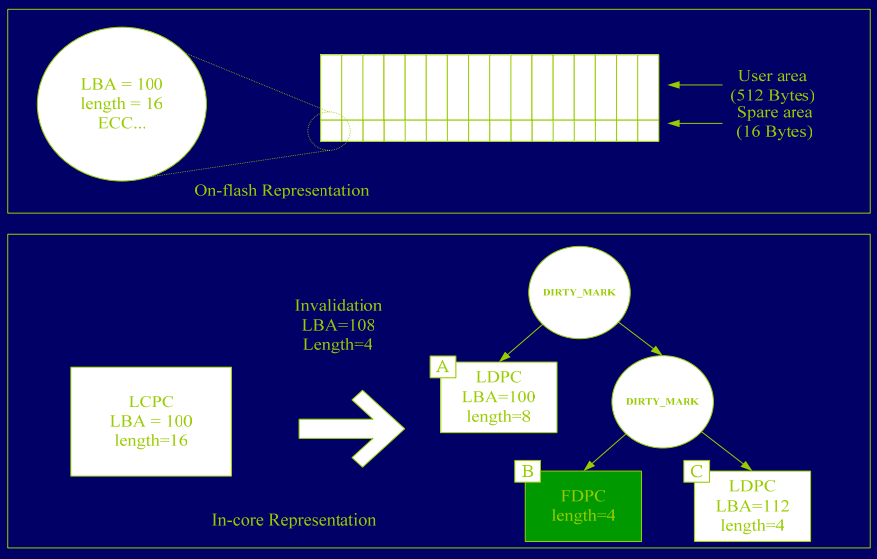
- Intelligent Multi-Granularity Address Translation Mechanisms!
  - Driver, firmware, or what levels?
  - Granularity Switching Mechanisms?
  - Space Utilization?
  - Garbage Collection Overheads?
  - System Initialization?

Large-Scale Flash – Cluster-Based Flash Management

- A physical clusters is a memory-resident object.
  - A PC describes the status of a set of contiguous flash memory pages.
  - Four statuses are indicated by a combination of a clean/dirty bit (C/D) and a free/live (F/L) bit.
  - Statuses of data stored on pages are maintained by PC's and committed to flash memory when needed.
- FDPCs (free and dirty PCs) correspond to available space.
  - It is different from FCPCs. They must be erased before they can be written.

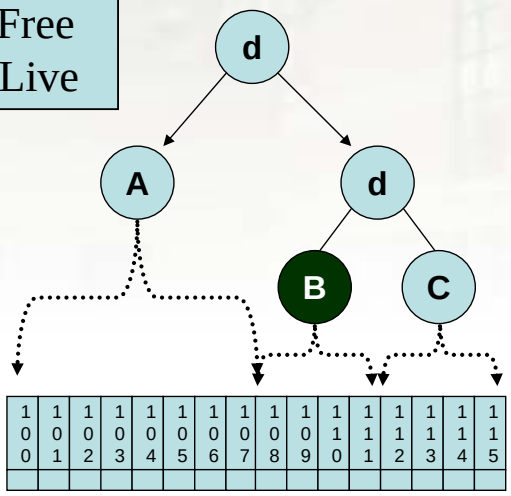
Large-Scale Flash – Cluster-Based Flash Management

- LDPC stores live data but might be involved in garbage collection!



Large-Scale Flash – Cluster-Based Flash Management

C: Clean
D: Dirty
F: Free
L: Live

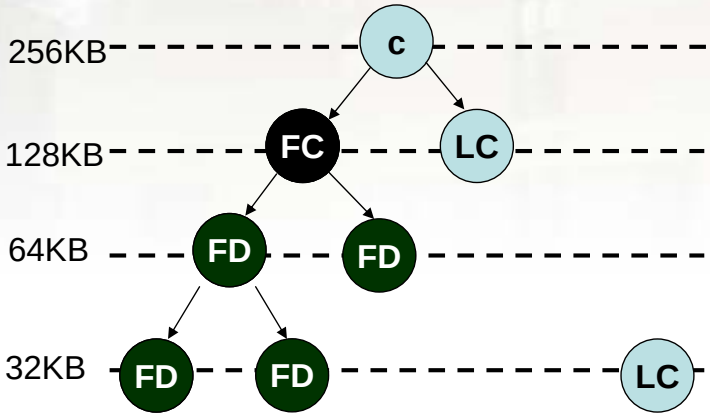


- d — A dirty_mark.
- A — An LDPC, LBA=100, length=8.
- B — An FDPC, length=4
- C — An LDPC, LBA=112, length=4

Large-Scale Flash – Cluster-Based Flash Management

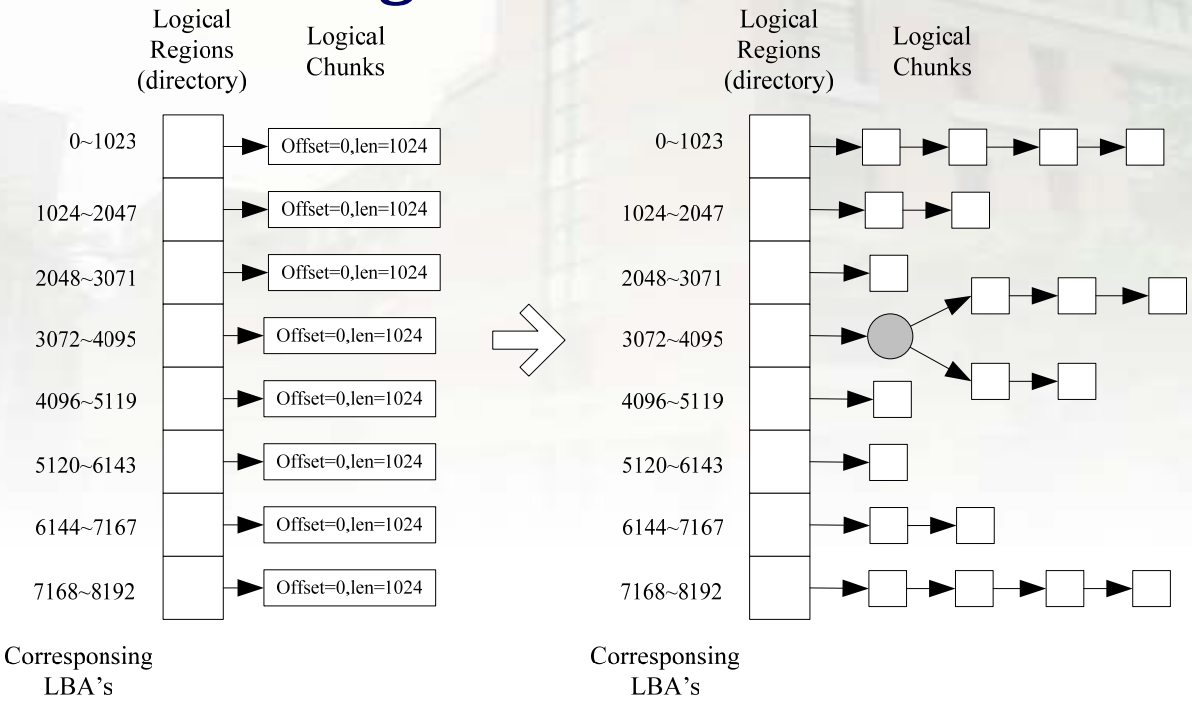
- Garbage Collection

C: Clean
D: Dirty
F: Free
L: Live



1. Data on LDPC is copied LDPC is invalidated as an FDPC
2. Two 32KB FDPC's are merged as one 64KB FDPC.
3. Two 64KB FDPC's are merged as one 128KB FDPC.
4. The 128KB FDPC is erased and becomes one FCPC.

Large-Scale Flash – Cluster-Based Flash Management



(a) The initial state of the hash table.

(b) The state after the processing of a sequence of updates.

Large-Scale Flash – Cluster-Based Flash Management

Fixed-sized granularity scheme (granularity = 1 page)

Total number of bytes written	26.3GB
Main-memory footprint size	**320MB**

Fixed-sized granularity scheme (granularity = 1 block)

Total number of bytes written	**48.9GB**
Main-memory footprint size	10MB

Our variable granularity sizes with Physical Clusters

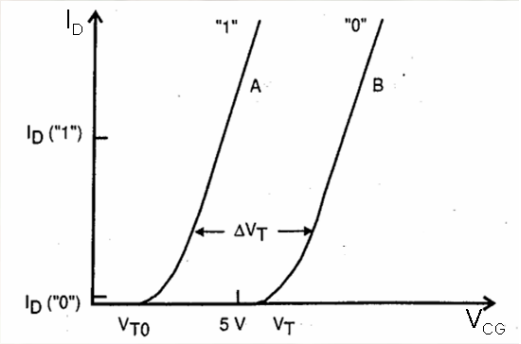
Total number of bytes written	26.18GB
Main-memory footprint size (peak)	22.6MB

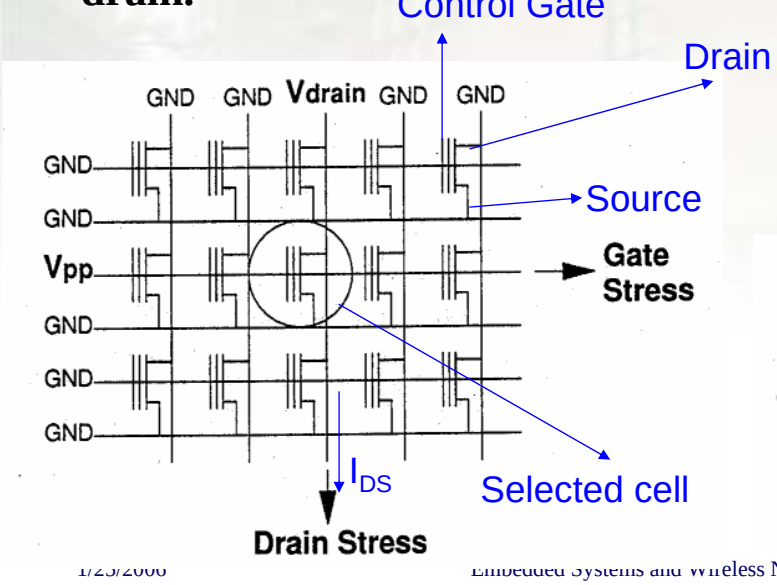
Agenda

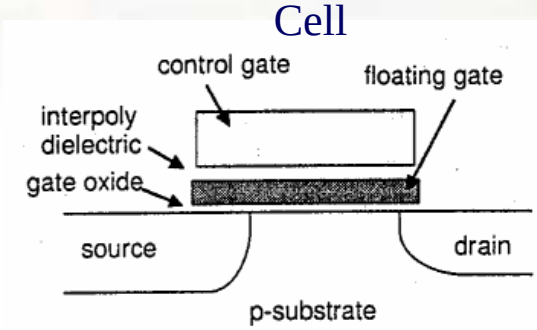
- Introduction
- Management Issues
- Performance vs Overheads
- Other Challenging Issues
- Conclusion

Challenging Issues – Reliability

- **Each Word Line is connected to control gates.**
- **Each Bit Line is connected to the drain.**

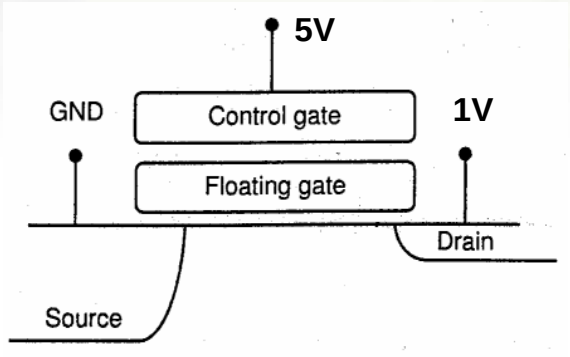


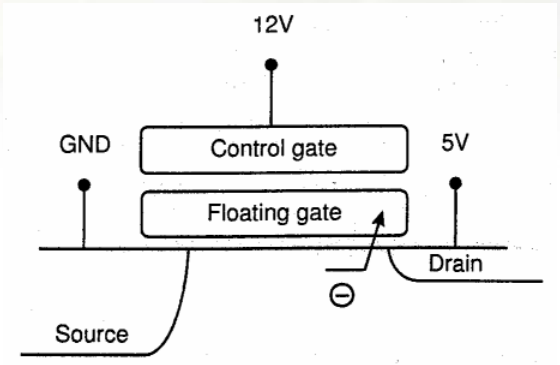




Challenging Issues – Reliability

- Read Operation
  - When the floating gate is not charged with electrons, there is current I_D (100*uA*) if a reading voltage is applied. ("1" state)
- Program Operation
  - Electrons are moved into the floating gate, and the threshold voltage is thus raised.





Challenging Issues – Reliability

- Over-Erasing Problems
  - Fast Erasing Bits → All of the cells connected to the same bit line of a depleted cell would be read as "1", regardless of their values.
- Read/Program Disturb Problems
  - DC erasing of a programmed cell, DC Programming of a non-programmed cell, drain disturb, etc.
  - Flash memory that has thin gate oxide makes disturb problems more serious!
- Data Retention Problems
  - Electrons stored in a floating gate might be lost such that the lost of electrons will sooner or later affects the charging status of the gate!

Challenging Issues – Observations

- The write throughput drops significantly after garbage collection starts!
- The capacity of flash-memory storage systems increases very quickly such that memory space requirements grows quickly.
- Reliability becomes more and more critical when the manufacturing capacity increases!
- The significant increment of flash-memory access numbers seriously exaggerates the Read/Program Disturb Problems!
- Wear-leveling technology is even more critical when flash memory is adopted in many system components or might survive in products for a long life time!

Conclusion

- Summary
  - Striping Issues
  - Hot-Data Identification
  - Scalability
- Challenging Issues
  - Scalability Technology
  - Reliability Technology
  - Customization Technology

Contact Information

- Professor Tei-Wei Kuo
  - ktw@csie.ntu.edu.tw
  - URL: http://csie.ntu.edu.tw/~ktw
  - Flash Research: http://newslab.csie.ntu.edu.tw/~flash/
  - Office: +886-2-23625336-257
  - Fax: +886-2-23628167
  - Address:

    Dept. of Computer Science & Information Engr.
    National Taiwan University, Taipei, Taiwan 106

Q & A