



# Embedded Operating System

Che-Wei Chang

[chewei@mail.cgu.edu.tw](mailto:chewei@mail.cgu.edu.tw)

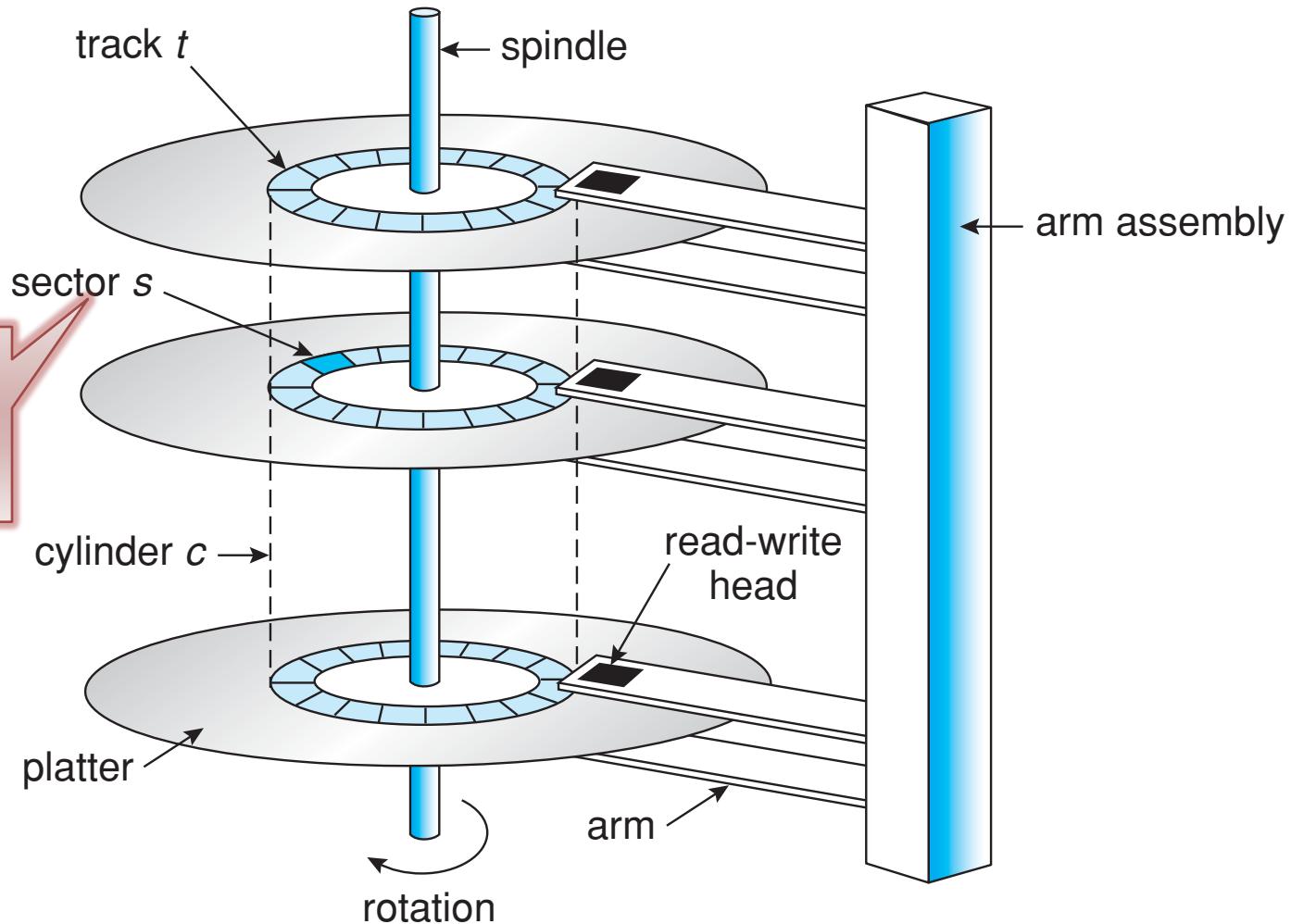
Department of Computer Science and Information  
Engineering, Chang Gung University



# Hard Drive Storage

# Moving-Head Disk Mechanism

The size of a sector is  
from 512B to 4KB



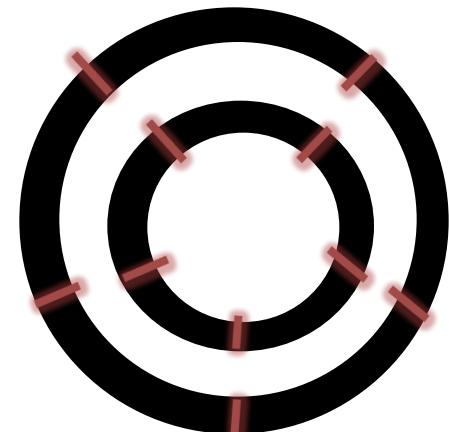
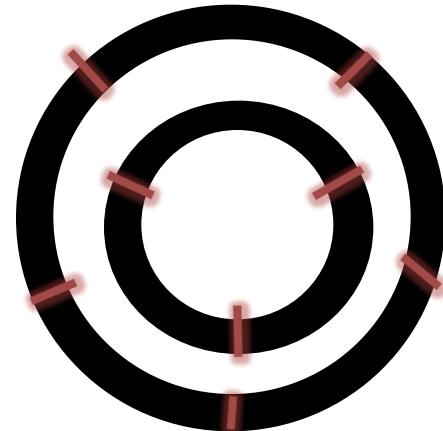
# Disk Structure

## Constant Linear Velocity (CLV)

- The outermost track typically hold 40 percent more sectors than the innermost track
- The drive increases its rotation speed as the head moves from the outer to the inner tracks
- The same rate of data moving is kept
- CD and DVD adopt this approach

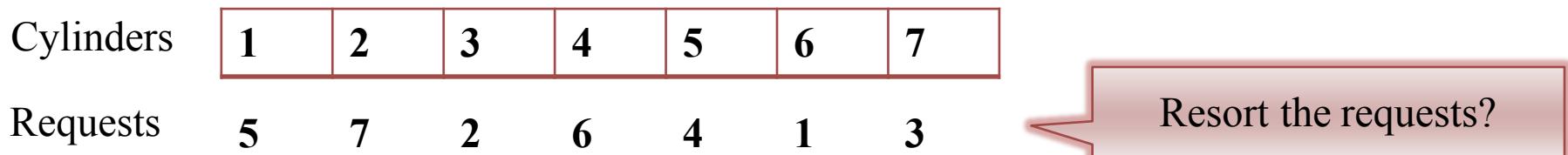
## Constant Angular Velocity (CAV)

- All tracks have the same number of sectors
- Tracks have different densities of sectors
- The same rate of data moving is kept
- HD adopts this approach



# Disk Scheduling

- ▶ The disk I/O request specifies several pieces of information:
  - Whether this operation is input or output
  - What the disk address for the transfer is
  - What the memory address for the transfer is
  - What the number of sectors to be transferred is
- ▶ When there are multiple request pending, a good disk scheduling algorithm is required
  - Fairness: which request is the most urgent one
  - Performance: sequential access is preferred



# Magnetic Disk Performance

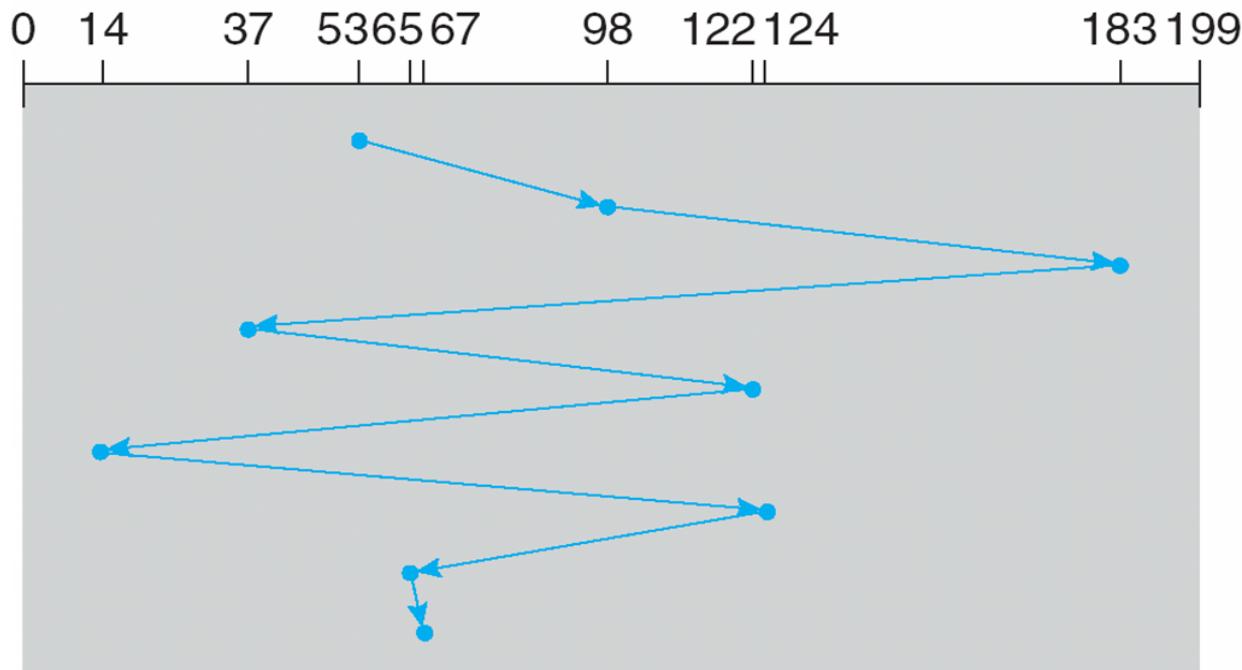
- ▶ Access Latency = Average access time = average seek time + average rotation latency
  - For fast disk  $3\text{ms} + 2\text{ms} = 5\text{ms}$
  - For slow disk  $9\text{ms} + 5.56\text{ms} = 14.56\text{ms}$
- ▶ Average I/O time = average access time + (amount to transfer / transfer rate) + controller overhead

# FCFS Scheduling

- ▶ FCFS: first come, first serve
- ▶ FCFS scheduling is fair but might with low throughput

queue = 98, 183, 37, 122, 14, 124, 65, 67

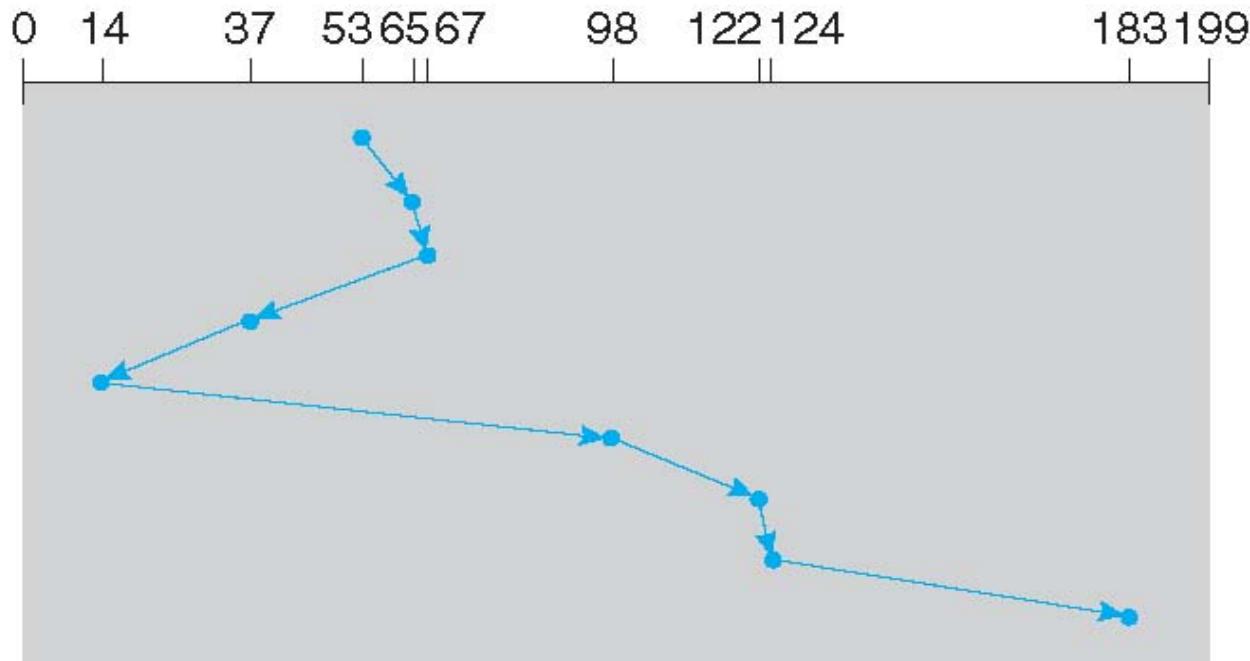
head starts at 53



# SSTF Scheduling

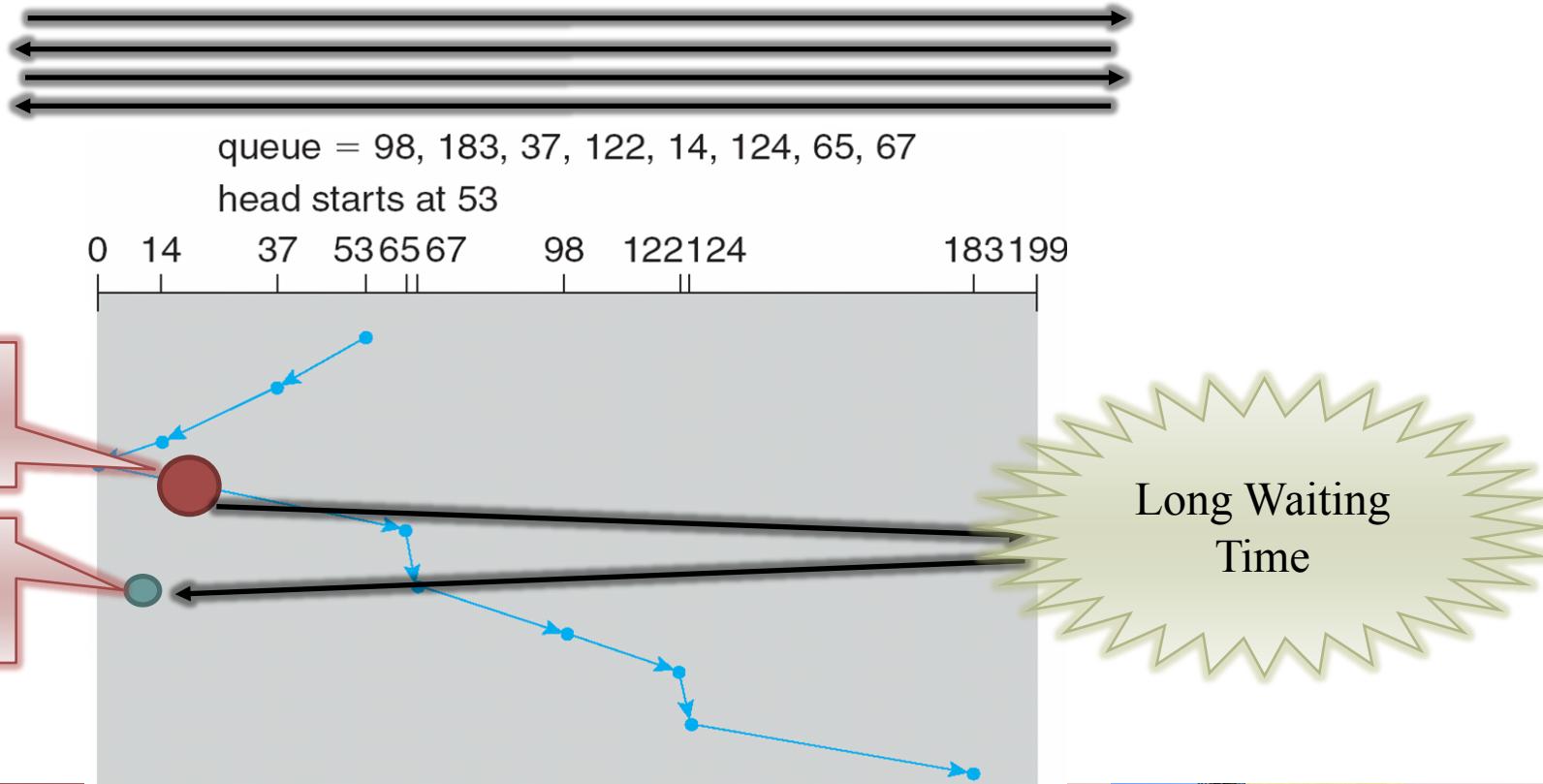
- ▶ SSTF: shortest seek time first
- ▶ SSTF scheduling serves the request with shortest seek time

queue = 98, 183, 37, 122, 14, 124, 65, 67  
head starts at 53



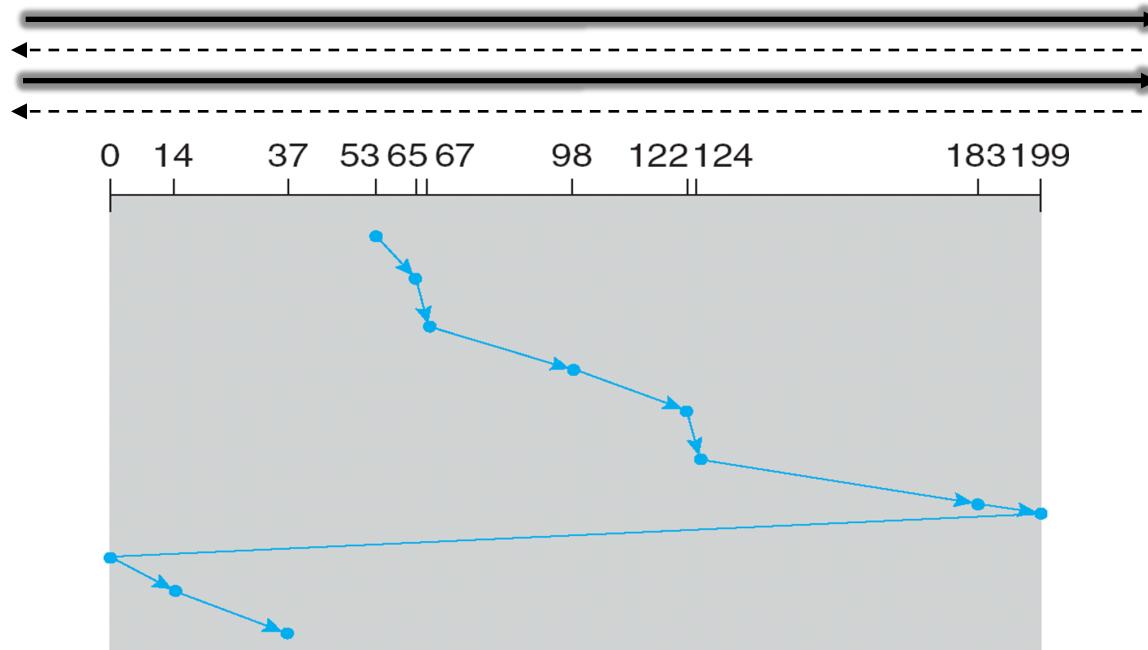
# SCAN Scheduling

- ▶ SCAN scheduling (also called the elevator algorithm) starts at one end and moves toward the other end



# C-SCAN Scheduling

- ▶ C-SCAN (Circular SCAN) scheduling starts at only one end and provides a more uniform wait time than SCAN scheduling

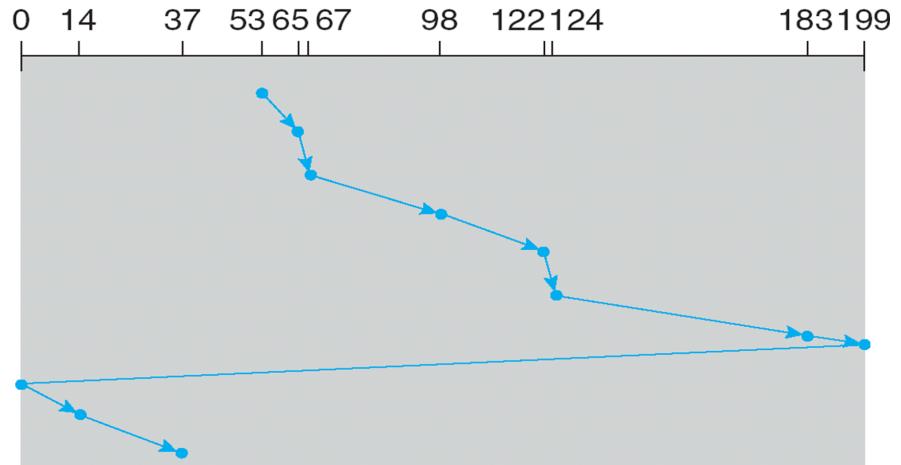


# LOOK and C-LOOK Scheduling

- ▶ LOOK scheduling starts at one end and moves toward the other end, and **looks for a request** before continuing to move in a given direction
- ▶ C-LOOK scheduling starts at only one end, and **looks for a request** before continuing to move in a given direction
- ▶ Arm only goes as far as the last request in each direction, then reverses direction immediately, **without first going all the way to the end of the disk**

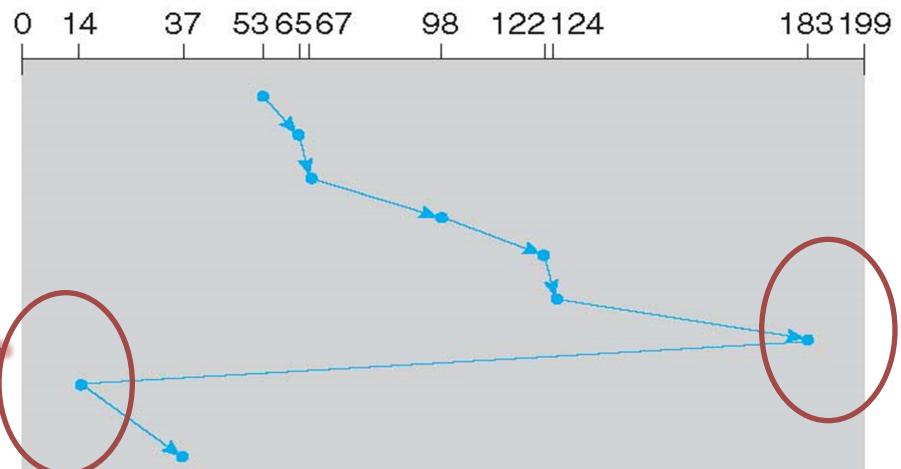
# Examples of C-SCAN and C-LOOK

C-SCAN



C-LOOK

Reduce the moving time



# Disk Management

- ▶ Low-level formatting, or physical formatting — Dividing a disk into sectors that the disk controller can read and write
  - Each sector can hold header information, plus data, plus error correction code (ECC)
  - Usually 512 ~ 4K bytes of data but can be selectable
- ▶ Partition the disk into one or more groups of cylinders, each treated as a logical disk
- ▶ Logical formatting — making a file system
  - To increase efficiency most file systems group blocks into clusters
    - Disk I/O done in blocks
    - File I/O done in clusters
- ▶ Raw disk access for apps that want to do their own block management, keep OS out of the way (databases for example)

# Bad Blocks

- ▶ A bad block: some bits of data in the block is corrupted
- ▶ Soft error: a bad block can be recovered by ECC
- ▶ Hard error: a bad block results in lost data
- ▶ Spared sectors are for bad block replacement
  - For example, one spared sector per 100 normal sector, let 97<sup>th</sup> block is a bad block
  - Sector sparing:
    - Use the spared sector to replace the 97<sup>th</sup> block
  - Sector slipping:
    - 97 → 98, 98 → 99, 99 → 100, 100 → spared sector



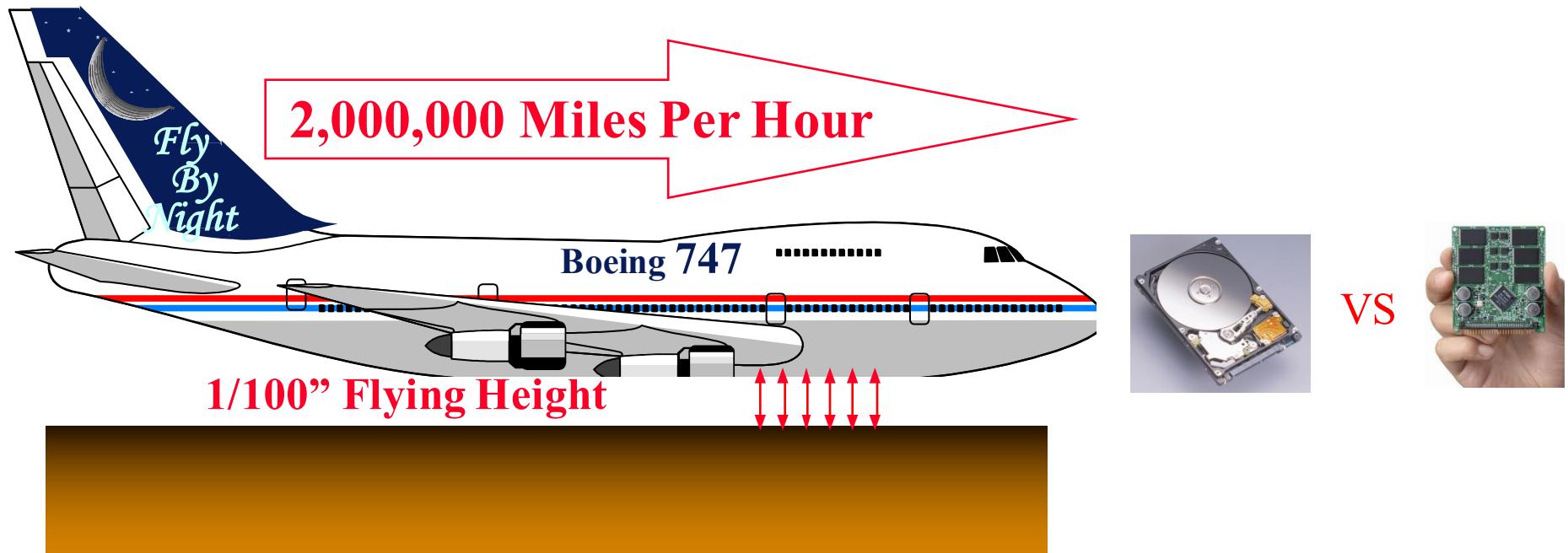
# Flash-Memory Storage

Reference: Prof. Tei-Wei Kuo, NTU and Dr. Yuan-Hao Chang, Academia Sinica

# Trends – Market and Technology

- ▶ Diversified Application Domains
  - Portable Storage Devices
  - Consumer Electronics
  - Industrial Applications
- ▶ Competitiveness in the Price
  - Dropping Rate and the Price Gap with HDDs
- ▶ Technology Trend over the Market
  - Improved density
  - Degraded performance
  - Degraded reliability

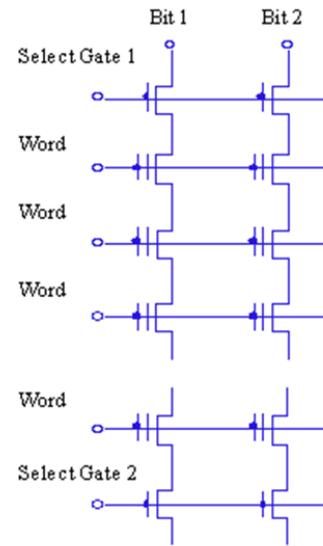
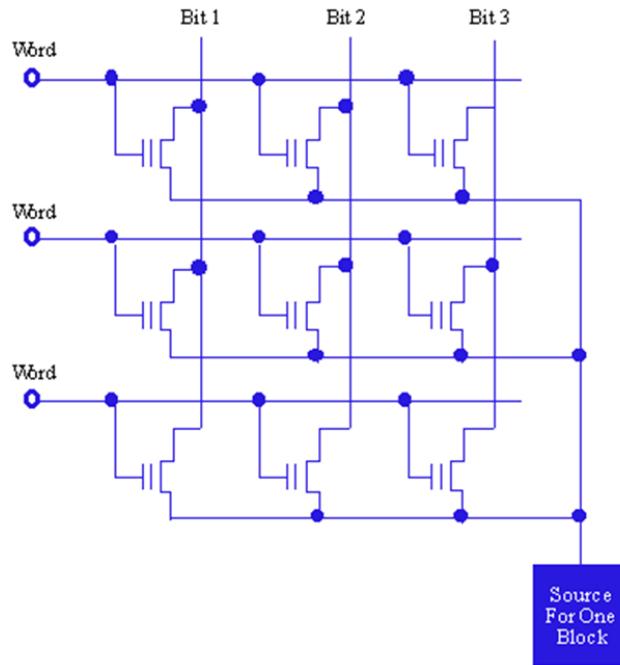
# Trends – Storage Media



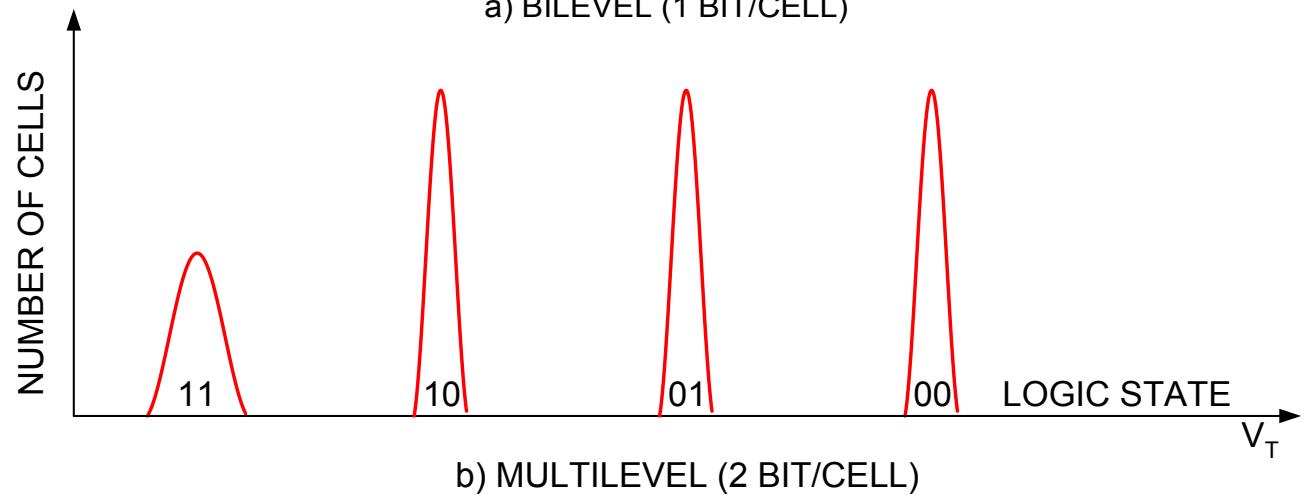
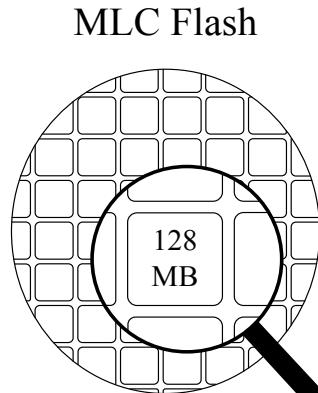
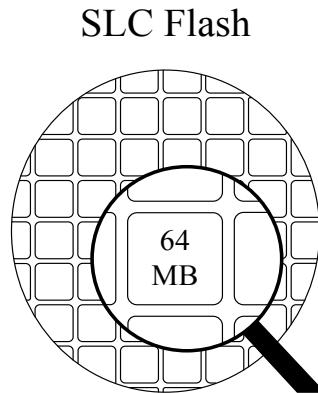
Source: Richard Lary, The New Storage Landscape: Forces shaping the storage economy, 2003.

# NOR and NAND Flash

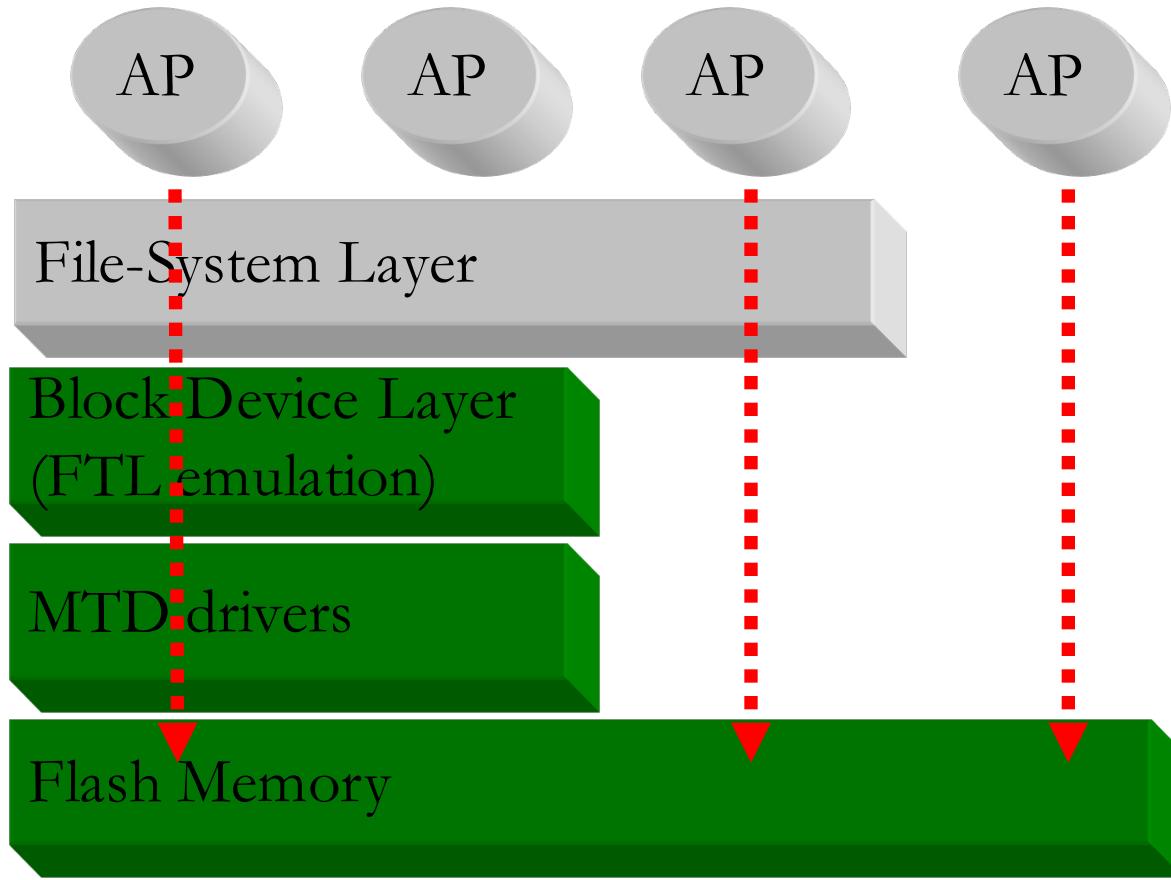
- ▶ NAND accesses each cell through adjacent cells, while NOR allows for individual access to each cell
- ▶ The cell size of NAND is almost half the size of a NOR cell



# Single-Level Cell (SLC) vs Multi-Level Cell (MLC) Flash



# System Architectures for Flash Management



# 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

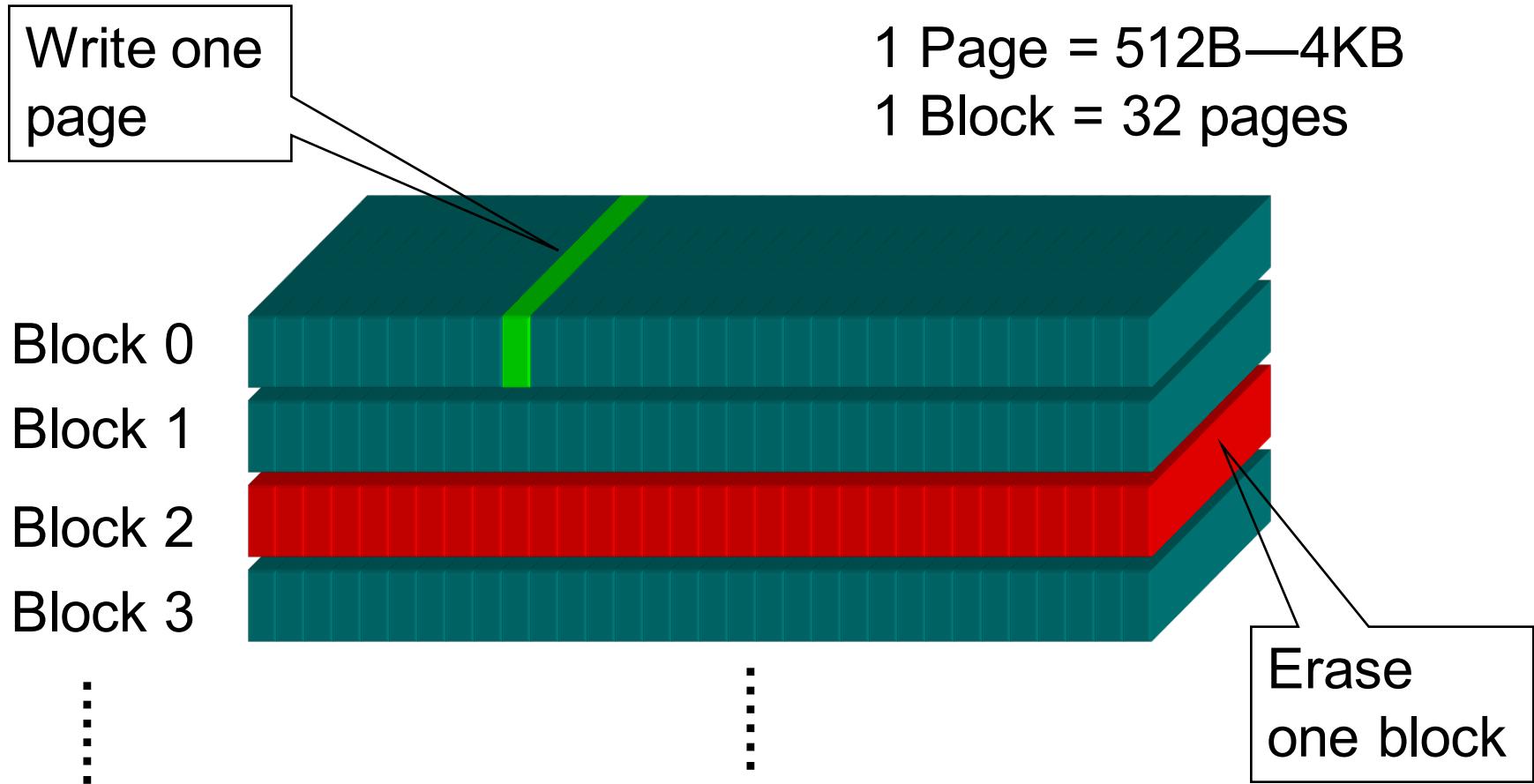


taobao.com

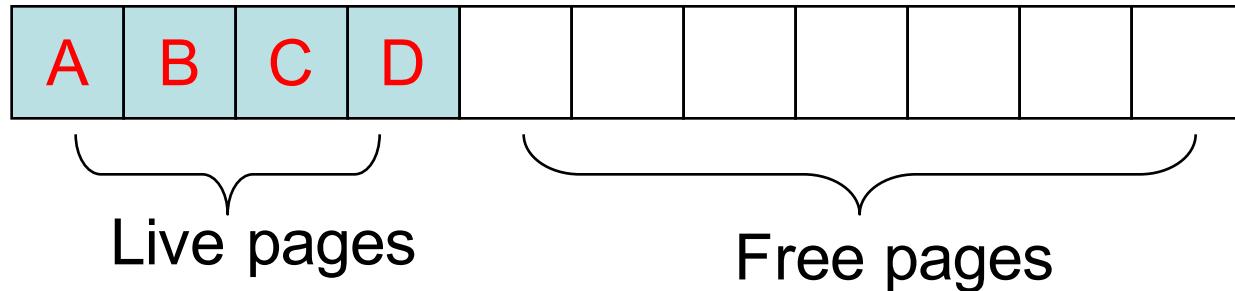
## ▶ Wear-Leveling

- Each block has a limited lifetime in erasing counts

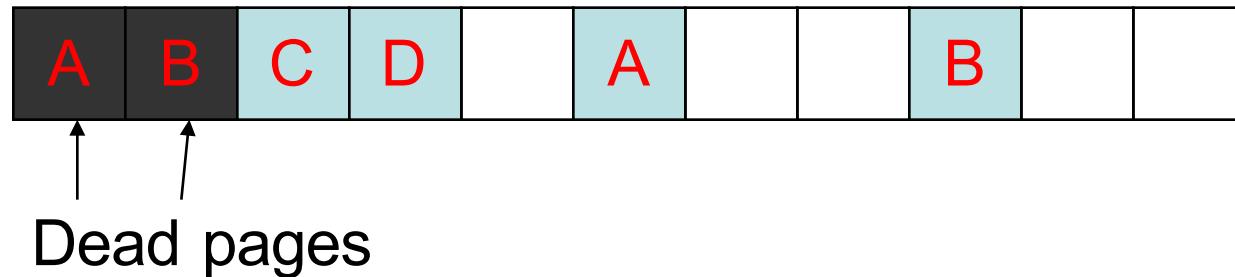
# Page Write and Block Erase



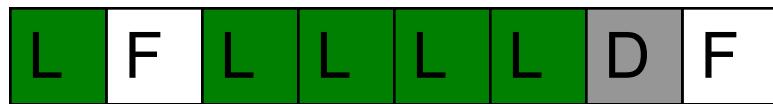
# Out-Place Update



Suppose that we want to update data A and B...



# Garbage Collection (1 / 3)



This block is to be recycled  
(3 live pages and 5 dead pages)

- A live page
- A dead page
- A free page

# Garbage Collection (2/3)



Live data are copied to somewhere else

- A live page
- A dead page
- A free page

# Garbage Collection (3/3)



The block is then erased

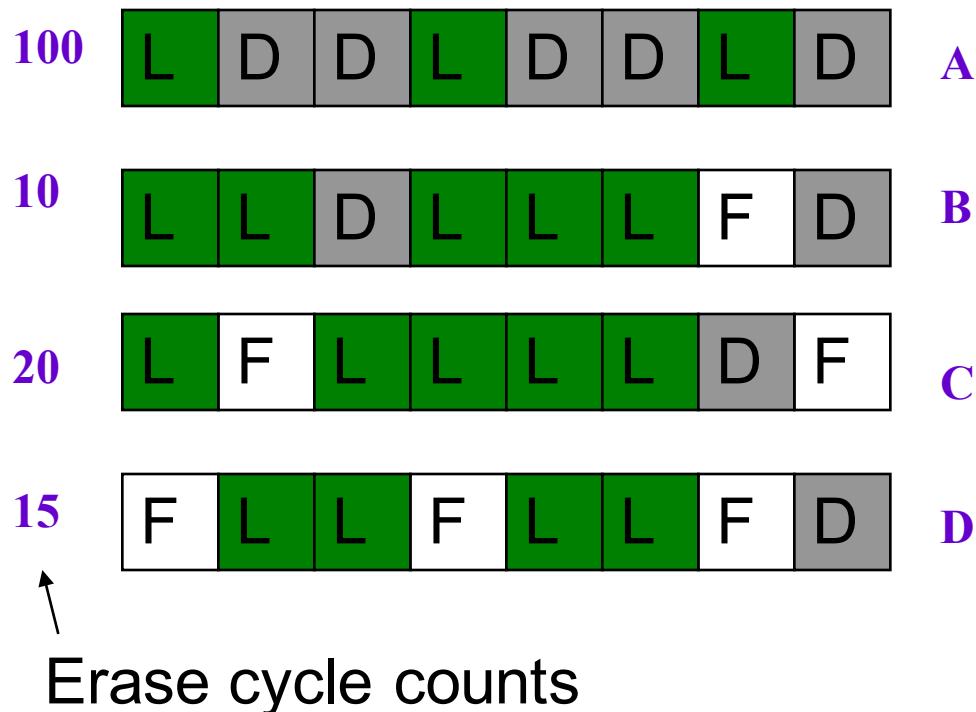


Overheads:

- live data copying
- block erasing

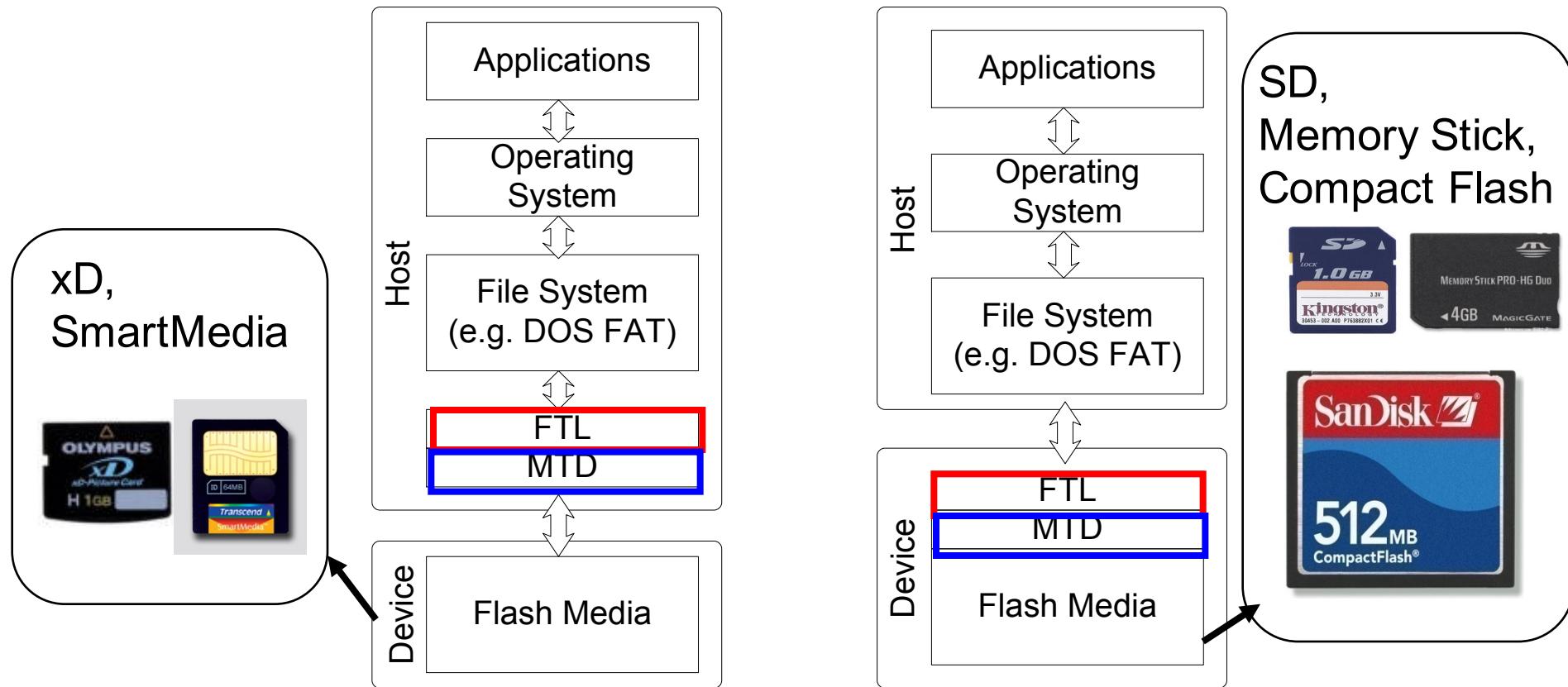
- A live page
- A dead page
- A free page

# Wear-Leveling



Wear-leveling might interfere with the decisions of the block-recycling policy

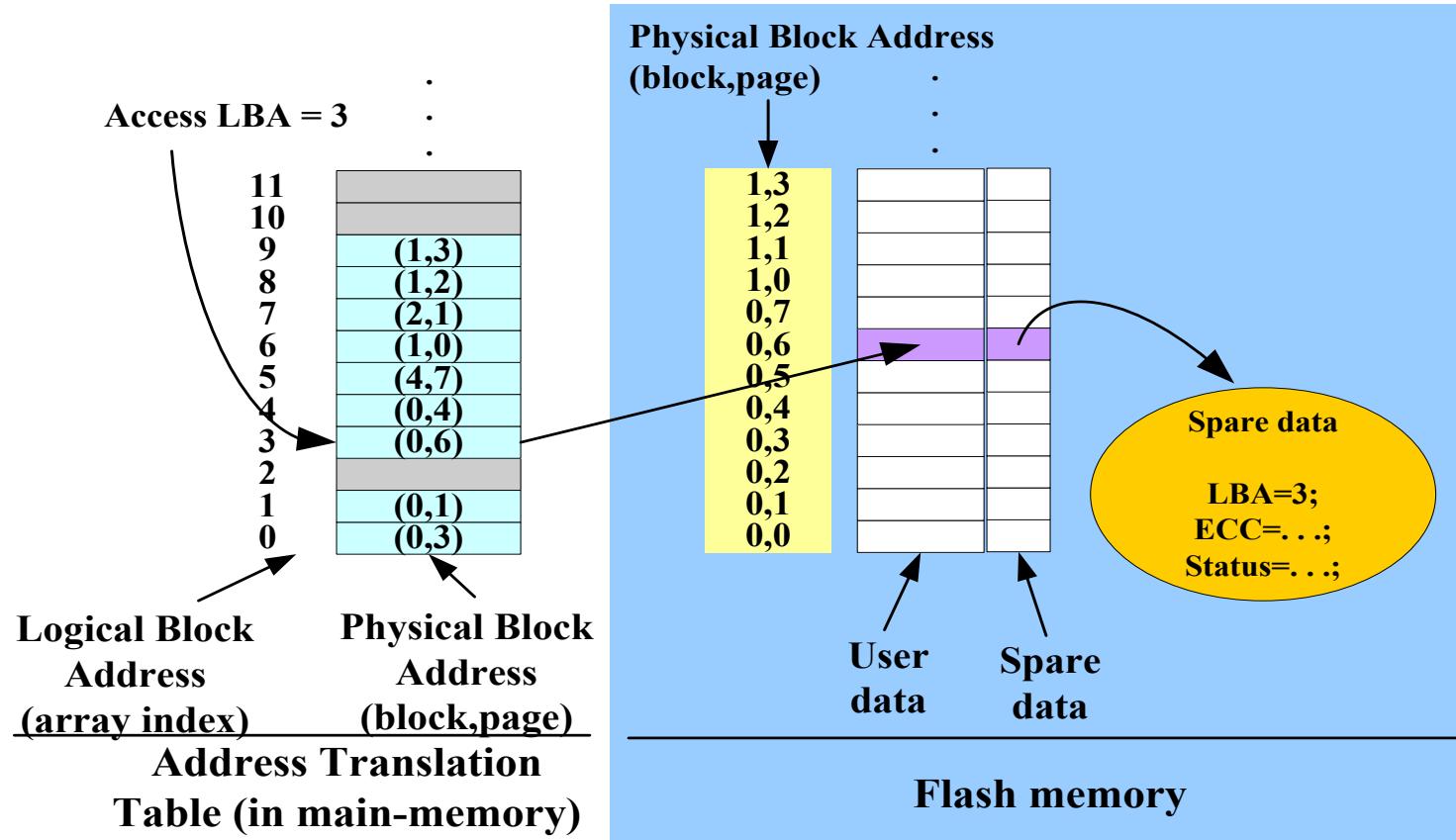
# Flash Translation Layer



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

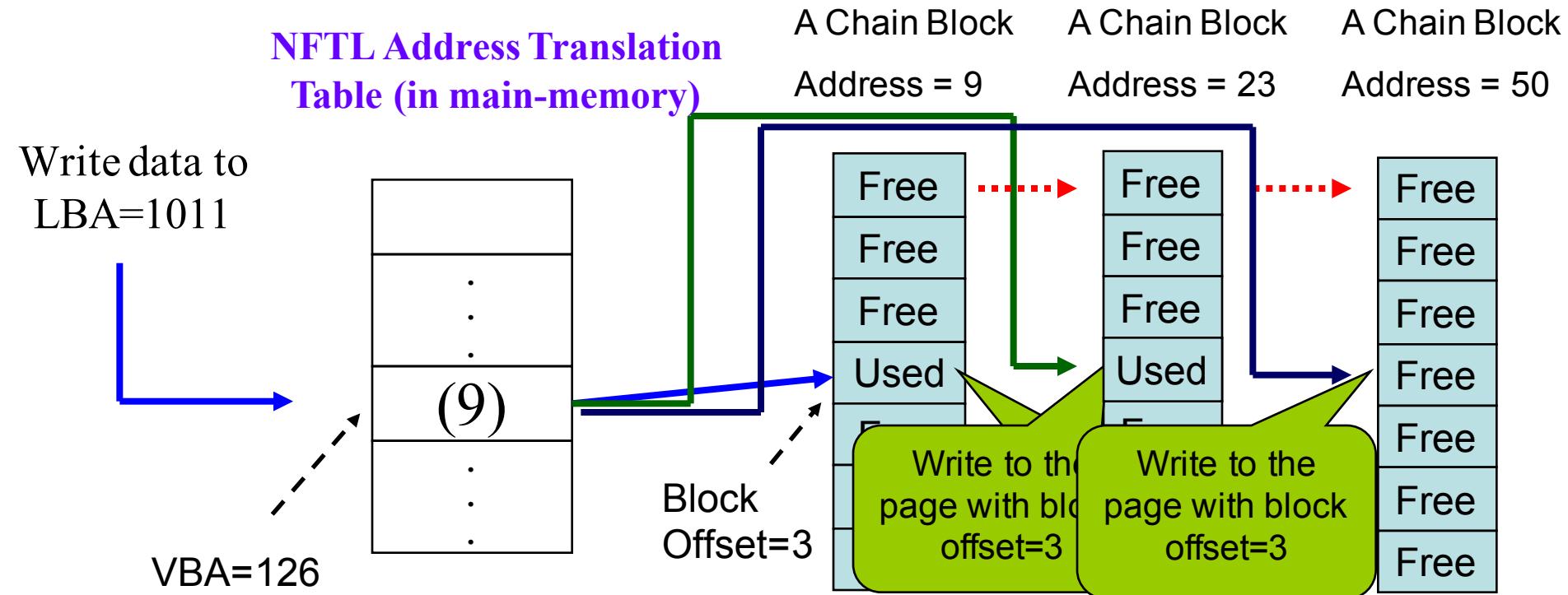
# Policies – FTL

- FTL adopts a page-level address translation mechanism



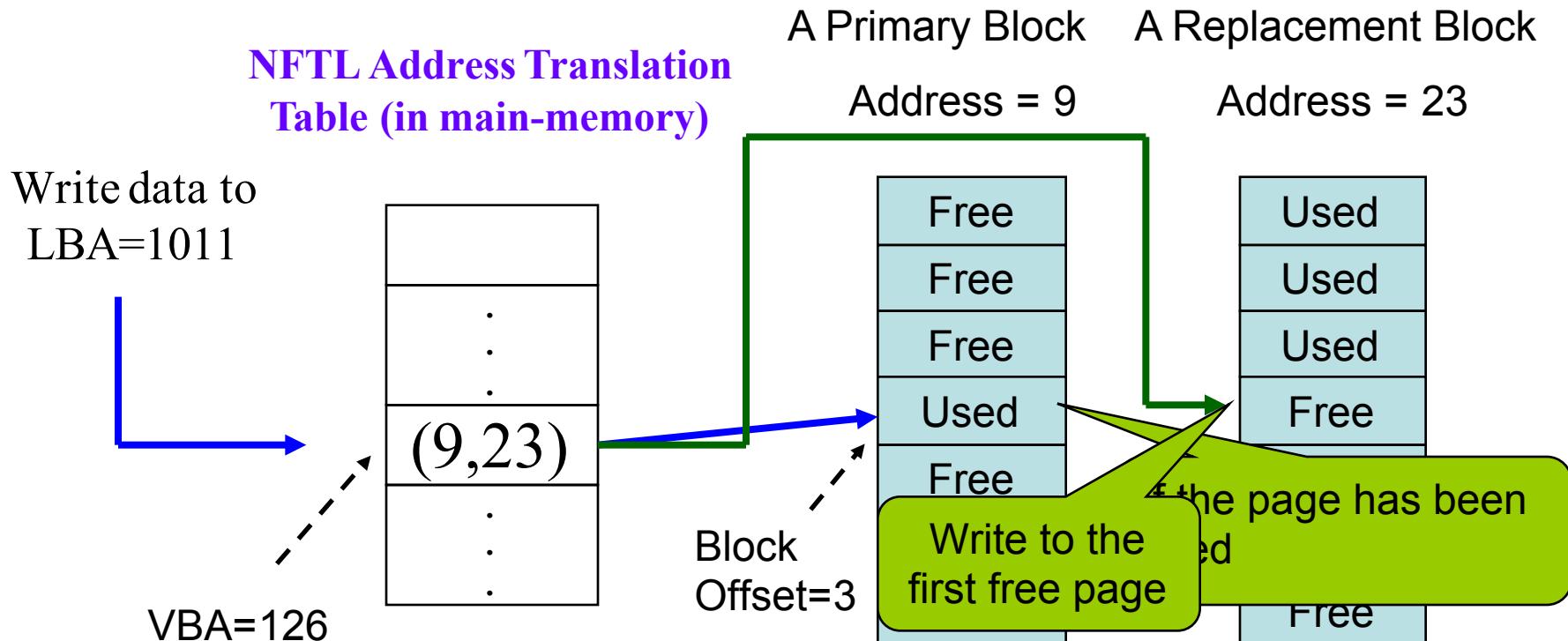
# Policies – NFTL (Type 1)

- ▶ 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 =  $1011 \% 8 = 3$



# Policies – NFTL (Type 2)

- ▶ 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



# Challenges and Research Topics of Flash Memory Designs

## ▶ Performance

- Reduce the overheads of Flash management
- Reduce the access time to data
- Reduce the garbage collection time

## ▶ Reliability

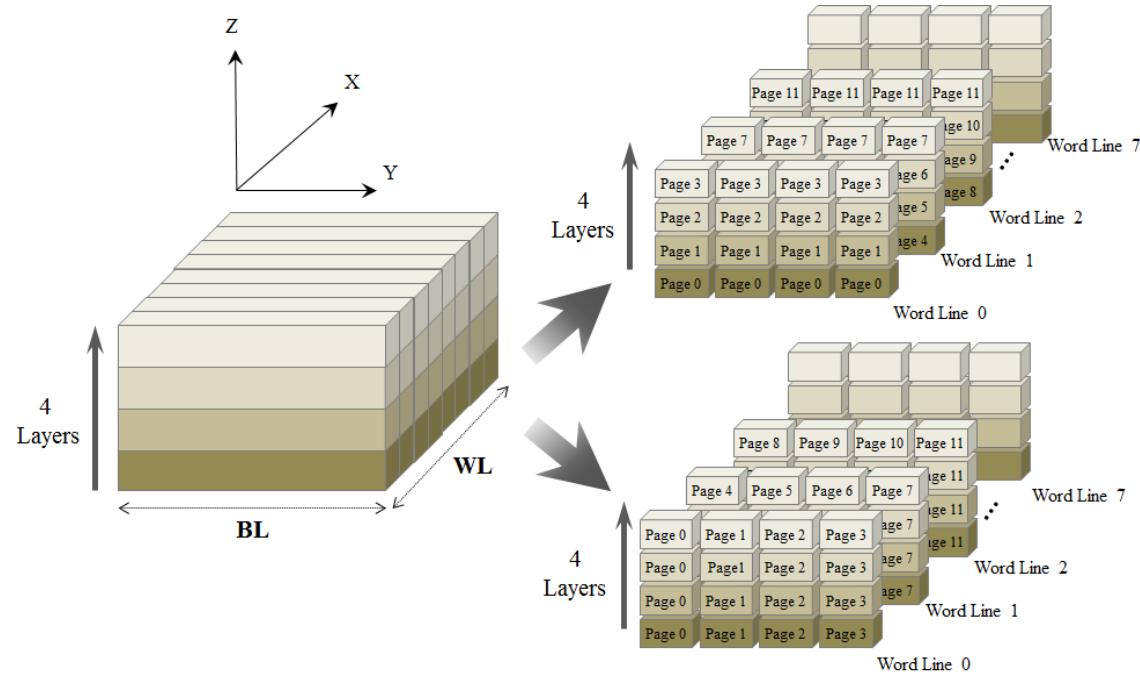
- Error correcting codes
- Log systems

## ▶ Endurance

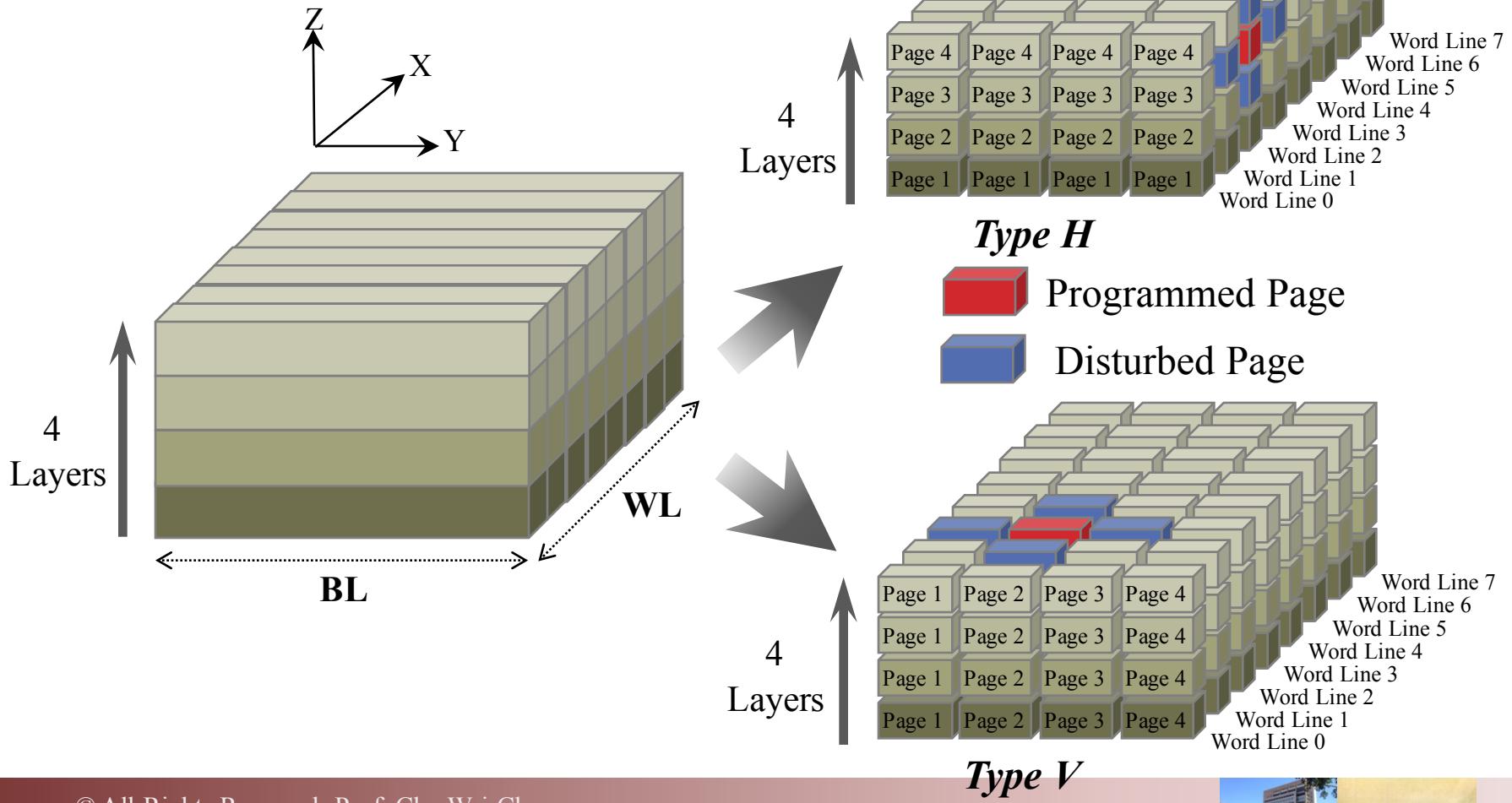
- Dynamic wear-leveling
- Static wear-leveling

# 3D Flash Memory

- ▶ 3D flash memory provides a good chance to further scale down the feature size and to reduce the bit cost.
  - Deliver very large storage space
  - Worsen program disturbance

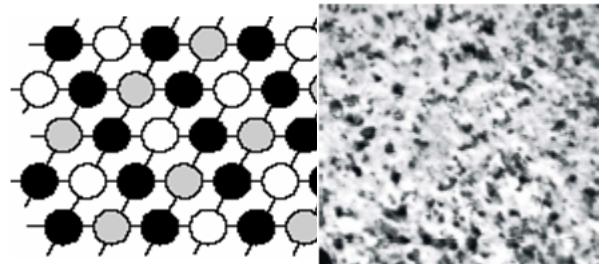
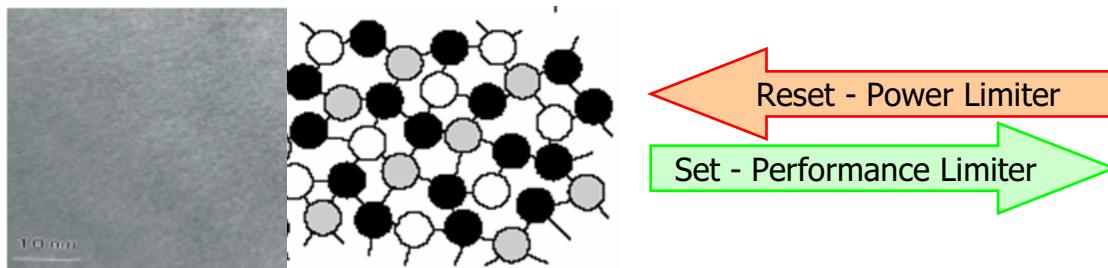
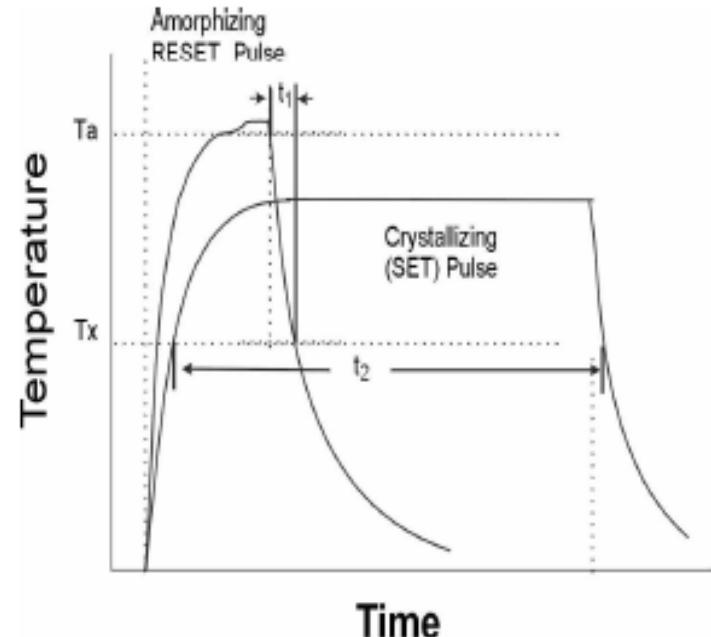


# Deteriorated Disturb on 3D Flash



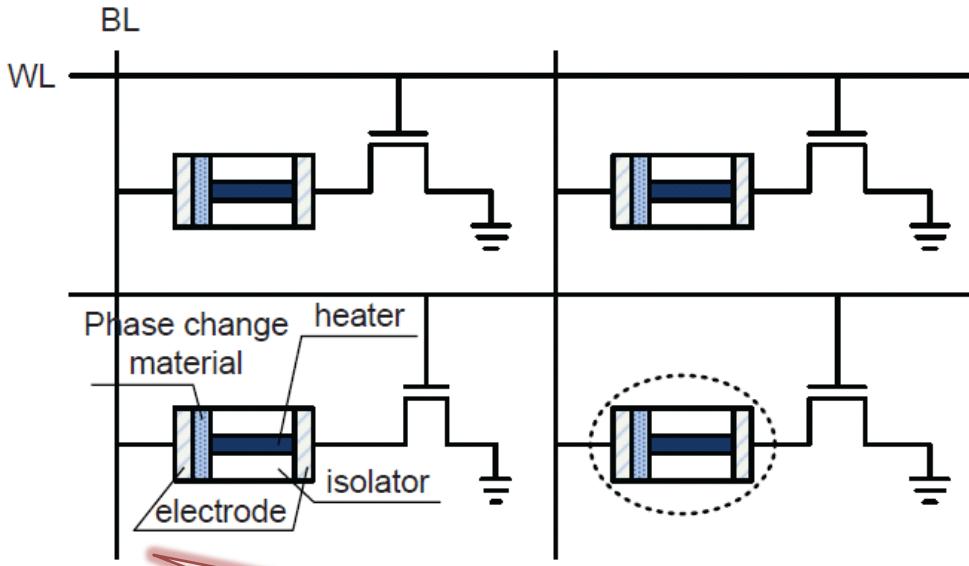
# Phase Change Memory (PCM)

- PCM is a non-volatile memory (NVRAM)
- PCM employs a reversible phase change in materials to store information.
- PCM exploits differences in the electrical resistivity of a material in different phases



# PCM Cell Array and Characteristics

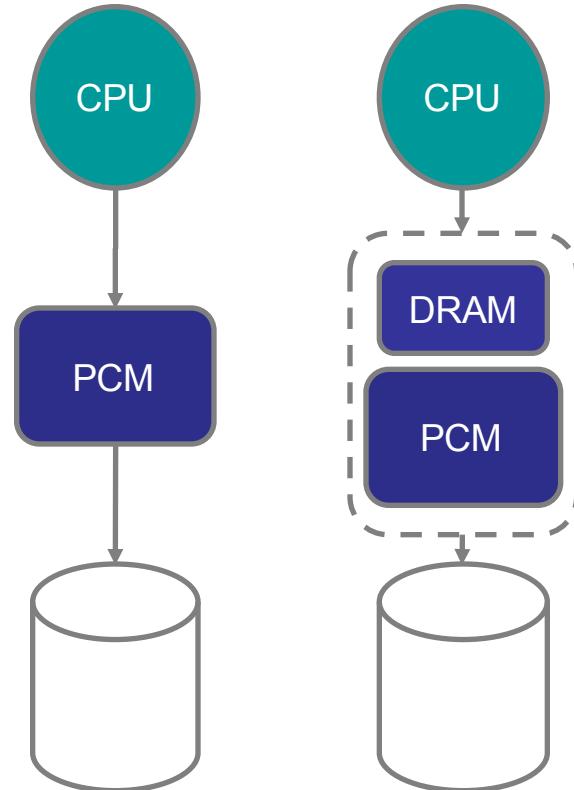
- ▶ Pros of PCM
  - Non-volatility
  - Bit-addressability
  - High scalability
  - No dynamic power
- ▶ Cons of PCM  
(compared to DRAM)
  - Low performance on writes
  - High energy consumption on writes
  - Low endurance



The read and write (SET and RESET) operations of a PCM cell require different current and voltage levels on the bitline, and take different amount of time to complete.

# PCM as Main Memory (1 / 2)

- ▶ Take advantage of its scalability and byte-addressability
- ▶ Challenges
  - Limited PCM endurance
  - Asymmetric read/write performance

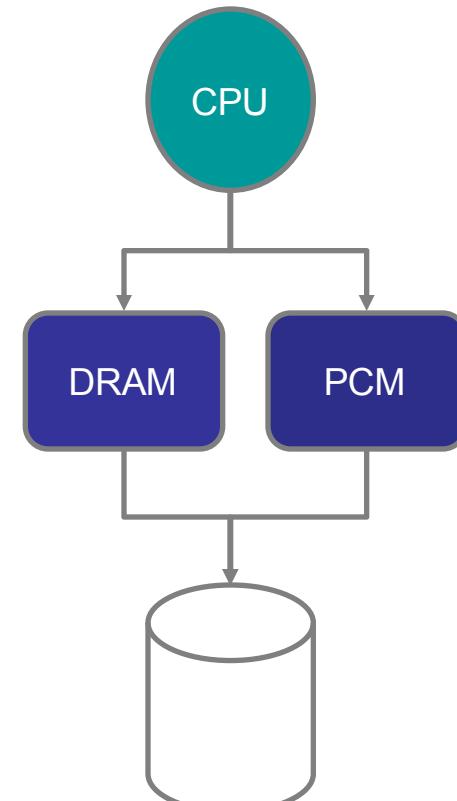


System with PCM

System with hybrid  
memory : DRAM as cache

# PCM as Main Memory (2/2)

- ▶ Take advantage of its non-volatility and byte-addressability
- ▶ Challenges:
  - What data should be in DRAM
  - What data should be in PCM
  - How to reuse data after power-off



System with hybrid memory

# PCM as Storage

- ▶ Take advantage of its non-volatility and high performance
- ▶ Challenges
  - Modern file systems have been built around the assumption that persistent storage is accessed via block-based interface
  - How to exploit its properties of persistent, byte-addressable memory



System with PCM

# PCM as Storage Class Memory

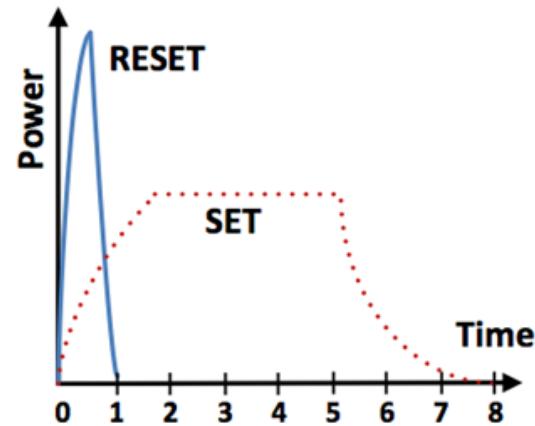
- ▶ IBM first proposed the idea of Storage Class Memory (SCM)
- ▶ PCM is the candidate of SCM
- ▶ SCM blurs the distinction between
  - Memory (fast, expensive, volatile) and
  - Storage (slow, cheap, non-volatile)



System with PCM as SCM

# Issues of Using PCM

- ▶ Write asymmetry
  - Reset
    - High instant power with short time
  - Set
    - Low power with long time
- ▶ Write latency
- ▶ Endurance issue

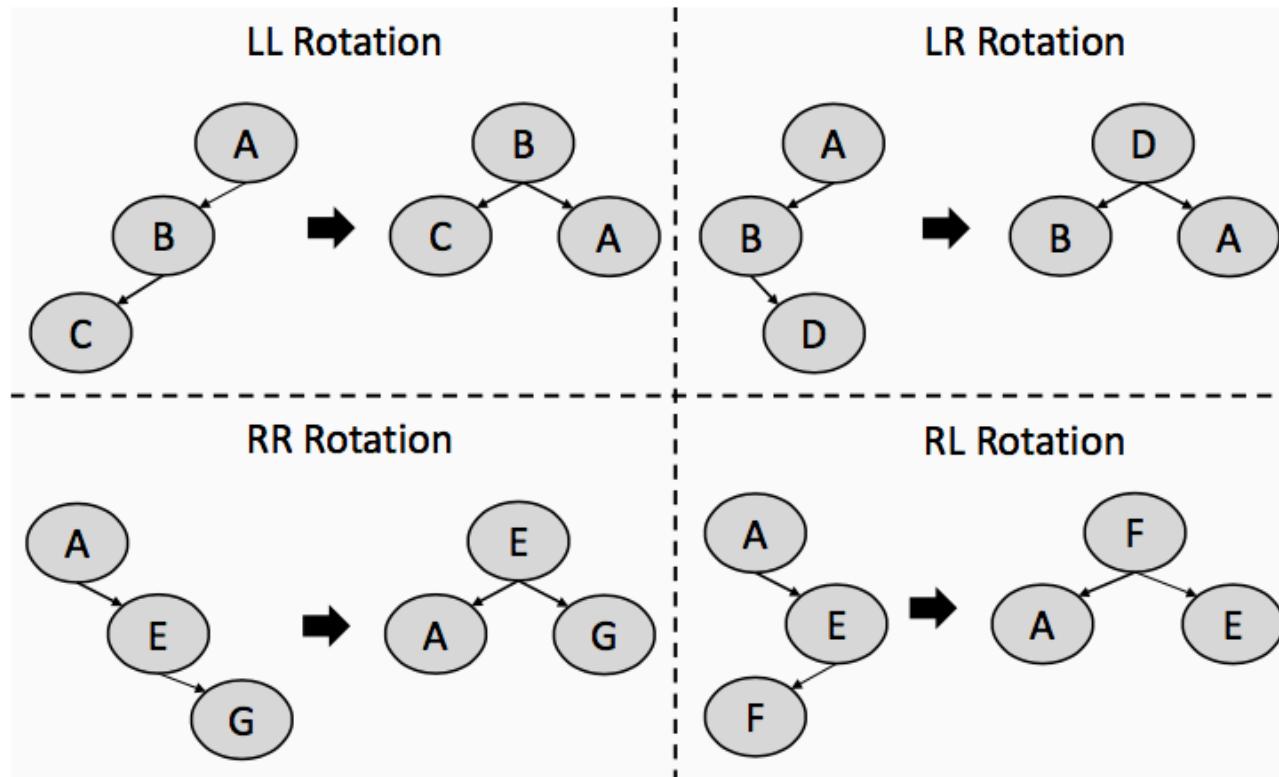


Types & Attributes	DRAM	PCM
Non-volatility	No	Yes
Bit alterability	Yes	Yes
Retention time	$\sim 60$ ms	$> 10$ years
Density	$20 - 32$ nm	$< 20$ nm
Write endurance	$> 10^{15}$ cycles	$10^6 - 10^8$ cycles
Write latency	$20 - 50$ ns	150 ns
Read latency	50 ns	50 ns

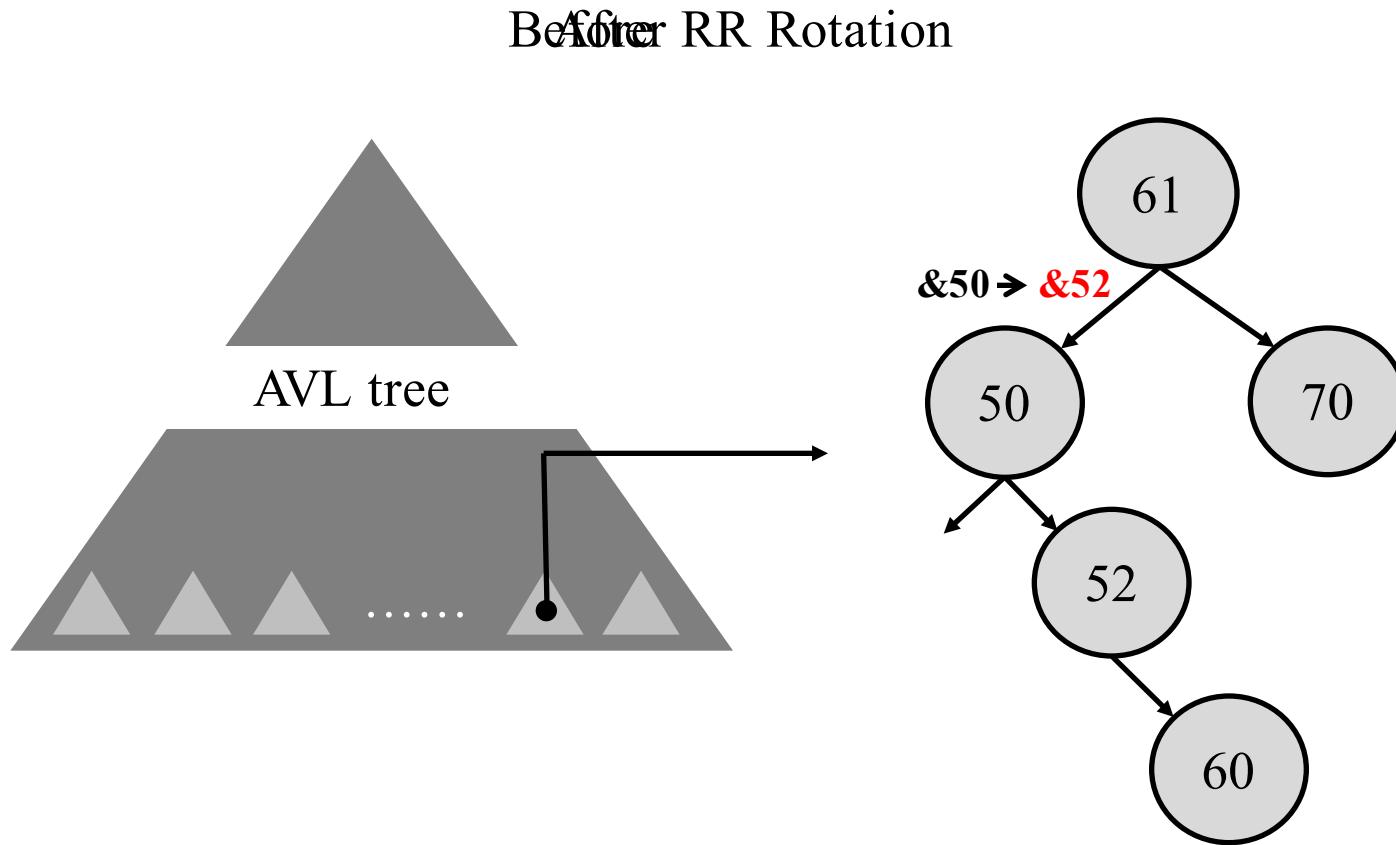
# Write Reduction on PCM

- ▶ Big/massive data applications demand extremely large main memory space for better performance
- ▶ PCM with low leakage power and high density is a promising candidate to replace DRAM
- ▶ Write endurance and latency are critical for using PCM
- ▶ Existing studies improve the write mechanism to handle given write patterns on PCM
- ▶ Why don't we improve fundamental data structures directly so as to generate more suitable write patterns for PCM

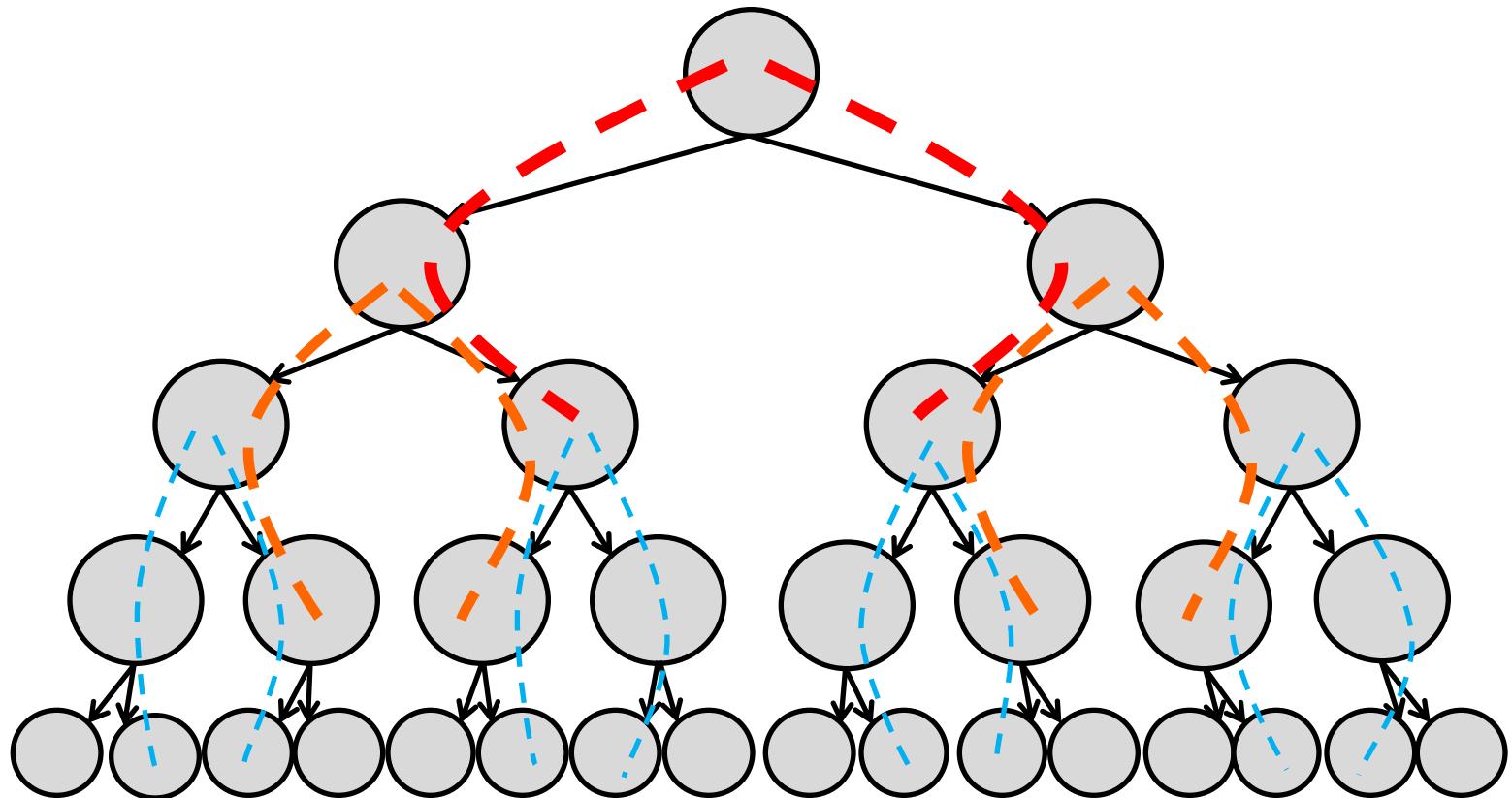
# Four Types of AVL Tree Rotations



# Relation among Nodes in an RR Rotation

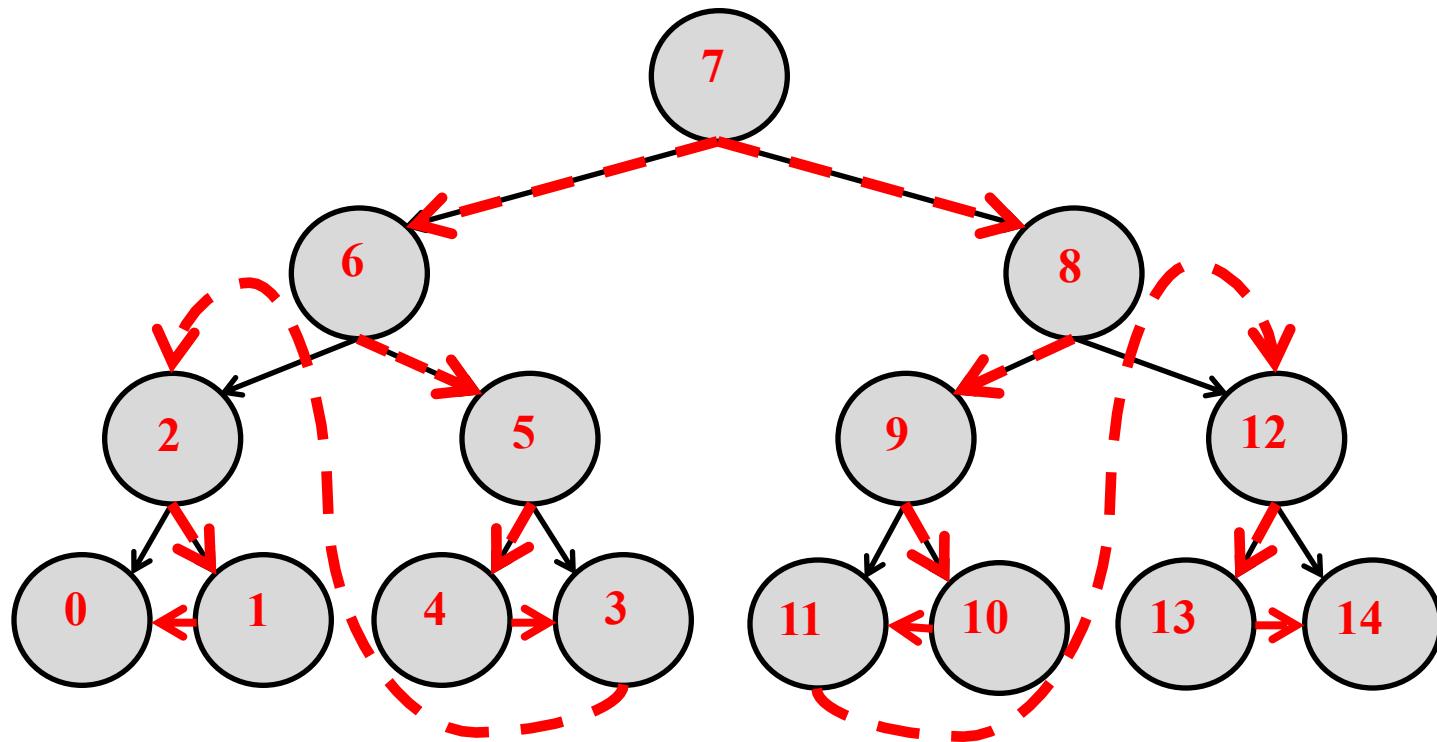


# Relation Binding of Tree Nodes



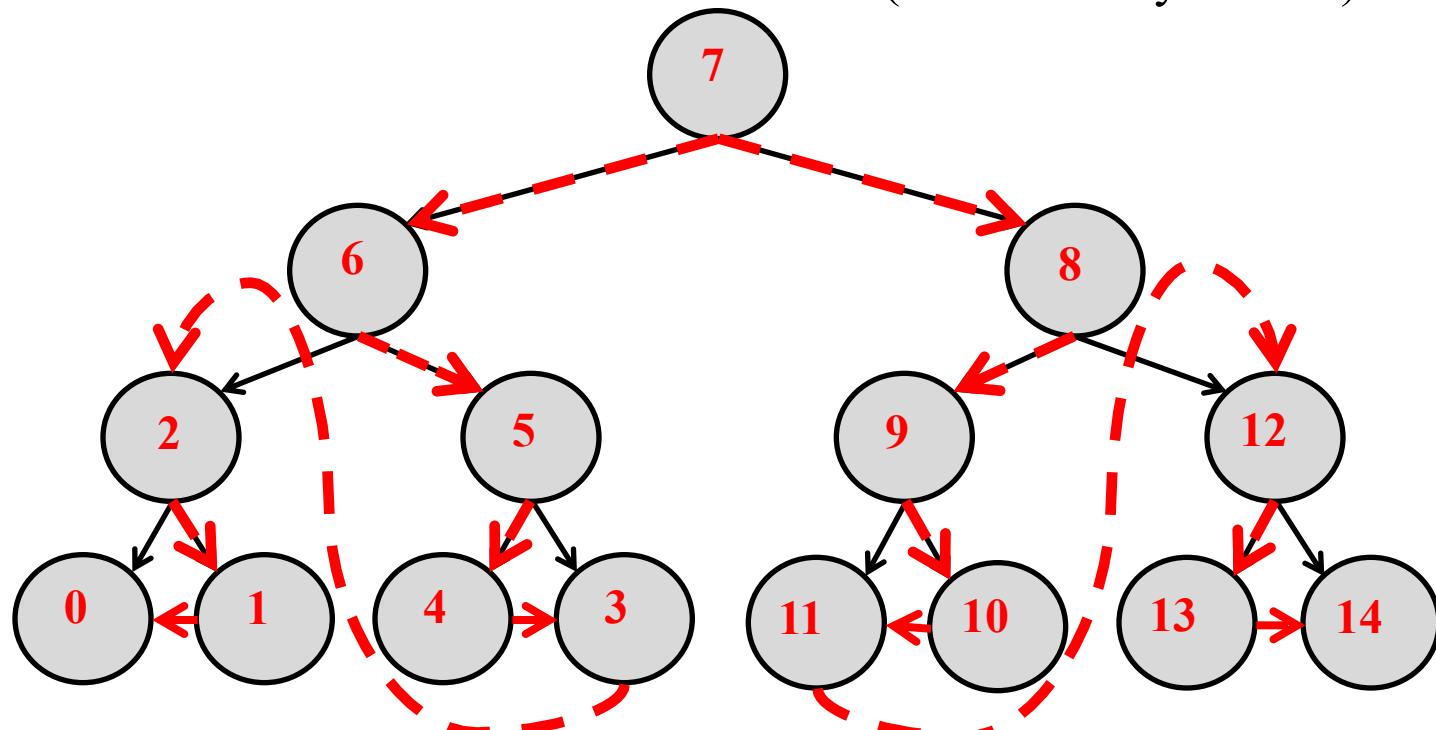
# Depth-First-Alternating Traversal (DFAT)

- ▶ A systematic approach for indexing all nodes, where nodes having stronger relations will be assigned closer indexes



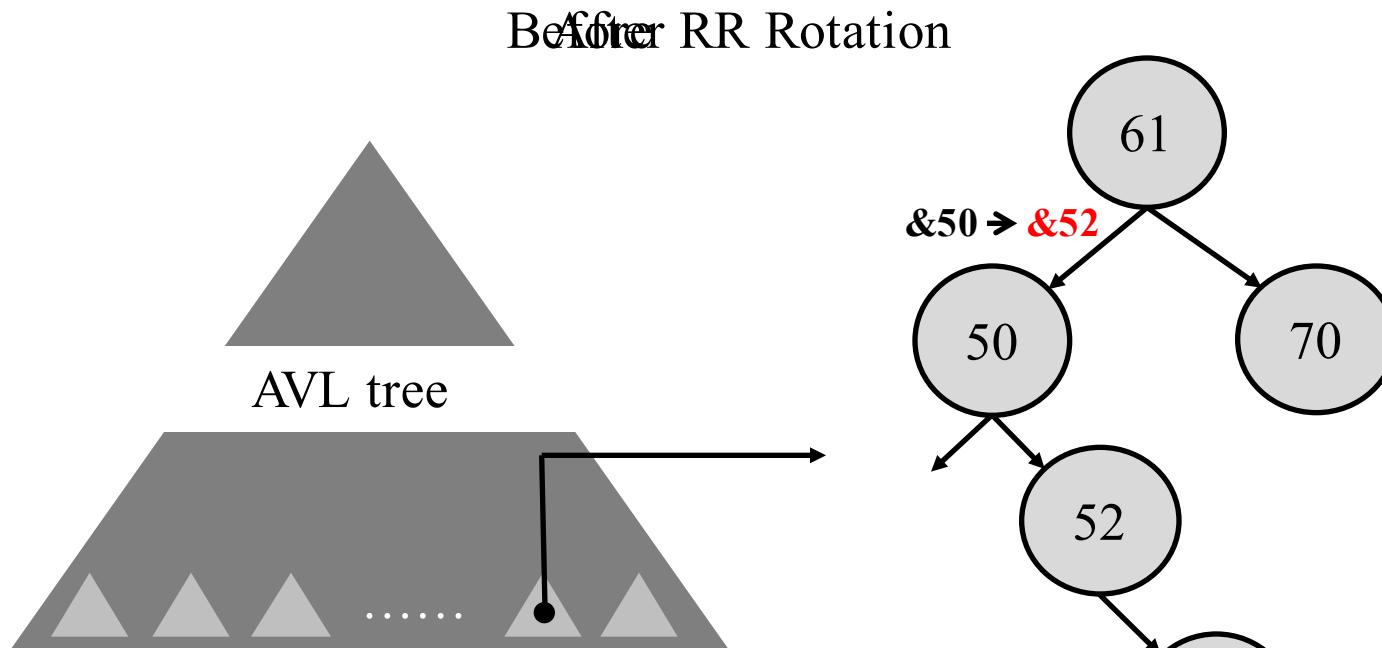
# Leveraging Gray Code on DFAT

- ▶ Gray code: An ordering of the binary numeral system such that two successive values have the shortest distance (differ in only one bit)



Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Gray Code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110

# An Example of Running DFAT with Gray Code



Key value	Binary code address	Gray code address
61	01111111111101	0100000000000011
70	01111111111110	0100000000000001
50	01111111111111	0100000000000000
52	1000000000000000	1100000000000000
60	1000000000000001	1100000000000001