Karlsruhe Institute of Technology

Betriebssysteme

15. Secondary-Storage Structure

Prof. Dr.-Ing. Frank Bellosa | WT 2016/2017

KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT) – OPERATING SYSTEMS GROUP

Secondary-Storage Structure

- Hard disk drives
- Solid state drive
- RAID structure
- Tertiary storage devices

Performance Levels of Storage

- Implicit/explicit movements between levels of storage hierarchy

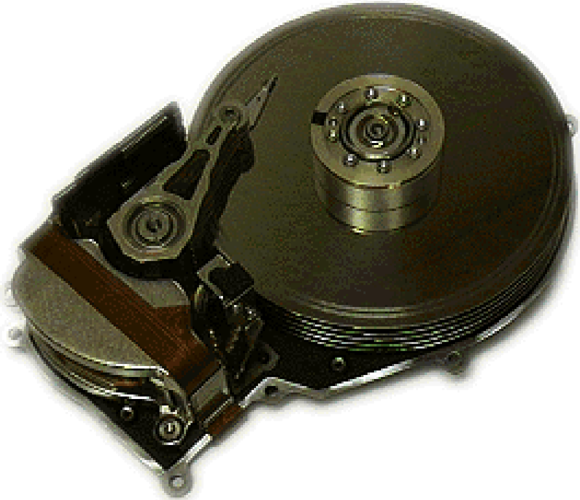
Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	$<$ 1KB	$<$ 64 MB	$<$ 64 GB	$>$ 100 GB
Implementation technology	custom memory multiport CMOS	on-/off-chip CMOS SRAM	DRAM PRAM STT-RAM	FLASH magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	5000 5.000.000
Bandwidth (MB/sec)	20.000 - 100.000	5000 - 10.000	1000 - 5000	20 - 150
Managed by	compiler (operating system)	hardware (operating system)	operating system	operating system
Backed by	cache	main memory	disk	DVD/tape

Anatomy of Hard Disk Drives

- Stack of magnetic platters
  - Rotate together on a central spindle @3,600-15,000 RPM
  - Drive speed drifts slowly over time
  - Can't predict rotational position after 100-200 revolutions

- Disk arm assembly
  - Arms rotate around pivot, all move together
  - Pivot offers some resistance to linear shocks
  - Arms contain disk heads–one for each recording surface
  - Heads read and write data to platters

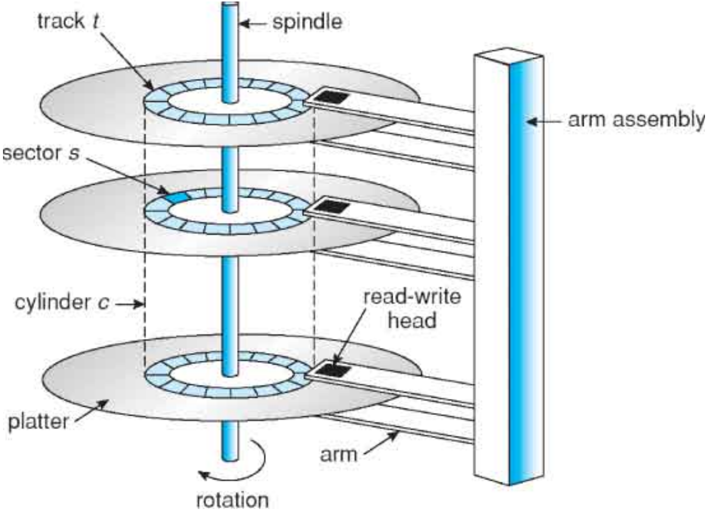
Hard Disk Drive



Storage on a Magnetic Platter

- Platters divided into concentric tracks
- A stack of tracks of fixed radius is a cylinder
- Heads record and sense data along cylinders
  - Significant fractions of encoded stream for error correction
- Generally only one head active at a time
  - Disks usually have one set of read-write circuitry
  - Must worry about cross-talk between channels
  - Hard to keep multiple heads exactly aligned
- Access time has two major components
  - Seek time is the time for the disk to move the heads to the cylinder containing the desired sector
  - Rotational delay is the additional time waiting for the disk to rotate the desired sector to the disk head

Cylinder, Tracks, Sectors



Disk Positioning System

- Move head to specific track and keep it there
  - Resist physical shocks, imperfect tracks, etc.
- A seek consists of up to four phases:
  - speedup– accelerate arm to max speed or half way point
  - coast–at max speed (for long seeks)
  - slowdown–stops arm near destination
  - settle– adjusts head to actual desired track
- Very short seeks dominated by settle time ($\sim$1 ms)
- Short (200-400 cyl.) seeks dominated by speedup
  - Accelerations of 40g

Seek Details

- Head switches comparable to short seeks
  - May also require head adjustment
  - Settles take longer for writes than for reads
  - If read strays from track, catch error with checksum, retry
  - If write strays, you've just clobbered some other track
- Disk keeps table of pivot motor power
  - Maps seek distance to power and time
  - Disk interpolates over entries in table
  - Table set by periodic "thermal recalibration"
  - But, e.g., $\sim$500 ms recalibration every $\sim$25 min bad for audio-/video-streaming
- "Average seek time" quoted can be many things
  - Time to seek 1/3 disk, 1/3 time to seek whole disk

Sectors

- Disk interface presents linear array of sectors (LBA)
  - Generally 512 bytes, written atomically (even if power failure)
- Disk maps logical sector #s to physical sectors (CHS)
  - Zoning–puts more sectors on longer tracks
  - Track skewing–sector 0 pos. varies by track
    - Improve sequential access speed
  - Sparing–flawed sectors remapped elsewhere
- OS doesn't know logical (LBA) to physical sector (CHS) mapping
  - Larger logical sector # difference means larger seek
  - Highly non-linear relationship, and depends on zone
  - OS has no info on rotational positions
  - Can empirically build table to estimate times

Disk Interface

- Controls hardware, mediates access
- Computer, disk often connected by bus (e.g., SCSI)
  - Multiple devices may content for bus
- Disk/interface features
  - Disconnect from bus during requests
  - Command queuing: Give disk multiple requests
    - Disk can schedule them using rotational information
  - Read-ahead into disk cache
    - Caching tracks to speed up sequential reads
    - Otherwise next block has to wait for a whole revolution
  - Write caching
    - But data not stable— not suitable for all requests

Disk Performance

- Placement & ordering of requests is a huge issue
  - Sequential I/O much, much faster than random
  - Long seeks much slower than short ones
  - Power might fail any time, leaving inconsistent state
- Must be careful about order to allow crash recovery
- Try to achieve contiguous accesses where possible
  - E.g., make big chunks of individual files contiguous
- Try to order requests to minimize seek times
  - OS can only do this if it has multiple requests to order
  - Requires disk I/O concurrency
  - High-performance apps try to maximize I/O concurrency

SSD: Principles of Operation

- Today, people are increasingly using flash memory
- Completely solid state (no moving parts)
  - Remembers data by storing charge
  - Lower power consumption and heat
  - No mechanical seek times to worry about
- Limited # overwrites
  - Blocks wear out after 10,000 (MLC) – 100,000 (SLC) erases
  - Requires flash translation layer (FTL) to provide wear leveling, so repeated writes to logical block don't wear out physical block
  - FTL can seriously impact performance
  - In particular, random writes are very expensive
- Limited durability
  - Charge wears out over time
  - Turn off device for a year, you can easily lose data

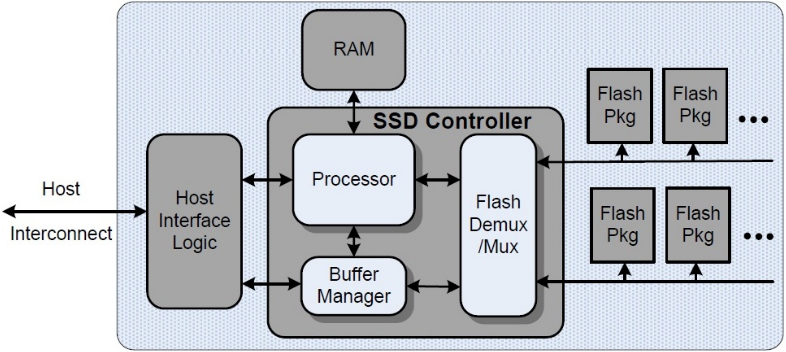
Types of Flash Memory

- NAND flash (most prevalent for storage)
  - Higher density (most used for storage)
  - Faster erase and write
  - More errors internally, so need error correction
- NOR flash
  - Faster reads in smaller data units
  - Can execute code straight out of NOR flash
  - Significantly slower erases
- Single-level cell (SLC) vs. Multi-level cell (MLC)
  - MLC encodes multiple bits in voltage level
  - MLC slower to write than SLC

NAND Flash Overview (typical device)

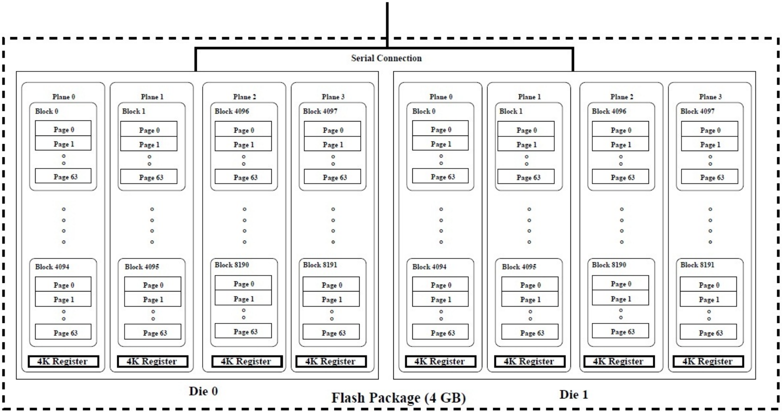
- Flash device has 2112-byte pages
  - 2048 bytes of data + 64 bytes metadata & ECC
- Blocks contain 64 (SLC) or 128 (MLC) pages
- Blocks divided into 2–4 planes
  - All planes contend for same package pins
  - But can access their blocks in parallel to overlap latencies
- Can read one page at a time
  - Takes 25 μs + time to get data off chip
- Must erase whole block before programming
  - Erase sets all bits to 1—very expensive (2 msec)
  - Programming pre-erased block requires moving data to internal buffer, then 200 (SLC)– 800 (MLC) μs

SSD Logic



RAM
SSD Controller
Host
Interconnect
Host
Interface
Logic
Processor
Buffer
Manager
Flash
Demux
/Mux
Flash
Pkg
Flash
Pkg
Flash
Pkg
Flash
Pkg

Flash Package



SSD Performance Optimization

- Spare block
  - The SSD controller always keeps a set of erased spare blocks
  - On a SSD block re-write the spare SSD-block is used for writing; the old SSD block is marked for erasure
  - Erasure takes place in background, if SSD device is idle
  - → SSD can tolerate the moderate modification rate without erase penalty
- `trim` Command
  - The operating system tells the SSD controller about deleted (unused) logical blocks
  - → Unused logical blocks don't have to be copied in case of a SSD block re-write (avoids Write Amplification)
  - → Unused logical blocks increase the pool of spare SSD blocks after garbage collection

Flash Characteristics

(see Gordon: Using Flash Memory to Build Fast, Power-efficient Clusters for Data-intensive Applications, Caulfield et al., APLOS 2009)

Parameter	SLC	MLC
Density Per Die (GB)	4	8
Page Size (Bytes)	2048+32	2048+64
Block Size (Pages)	64	128
Read Latency (μs)	25	25
Write Latency (μs)	200	800
Erase Latency (μs)	2000	2000
40MHz Read b/w (MB/s)	75.8	75.8
Program b/w (MB/s)	20.1	5.0
133MHz Read b/w (MB/s)	126.4	126.4
Program b/w (MB/s)	20.1	5.0

Improvements for Disk-I/O

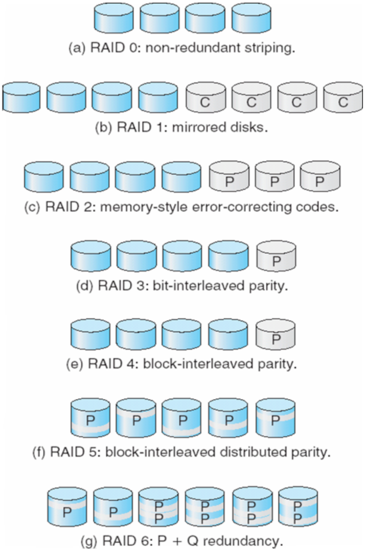
- Analysis:
  data rate of a disk $\ll$ data rate of CPU or RAM
- Idea:
  - Use multiple disks to parallelize disk-I/O
  - Provide a better disk availability
  - Instead of 1 single large expensive disk (**SLED**) use

$\Rightarrow$ RAID = redundant array of independent disks
(originally: redundant array of *inexpensive* disks)

RAID (Redundant Array of Inexpensive Disks)

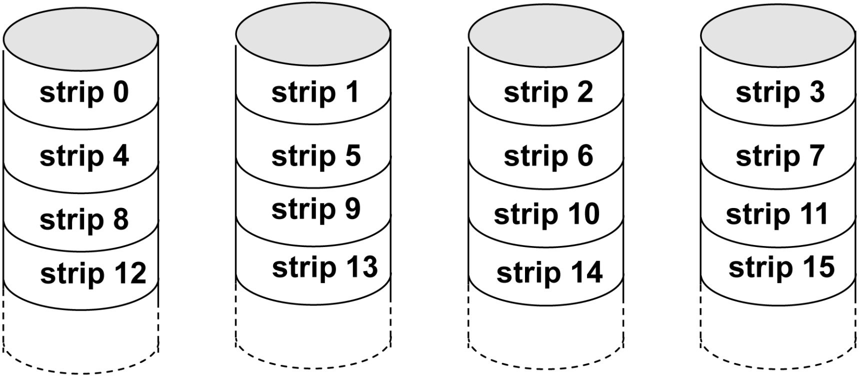
- Multiple disk drives provide high storage volume and performance with improved reliability compared to a large expensive disk (**SLED**)
- RAID schemes improve performance and reliability of the storage system by storing redundant data.
  - *Mirroring or shadowing* keeps duplicate of each disk.
  - *(Bit-/Byte-/Block-)) interleaved parity* used much less redundancy.
- Disk stripping uses a group of disks as one storage unit.
- RAID is arranged into six different levels.

RAID levels



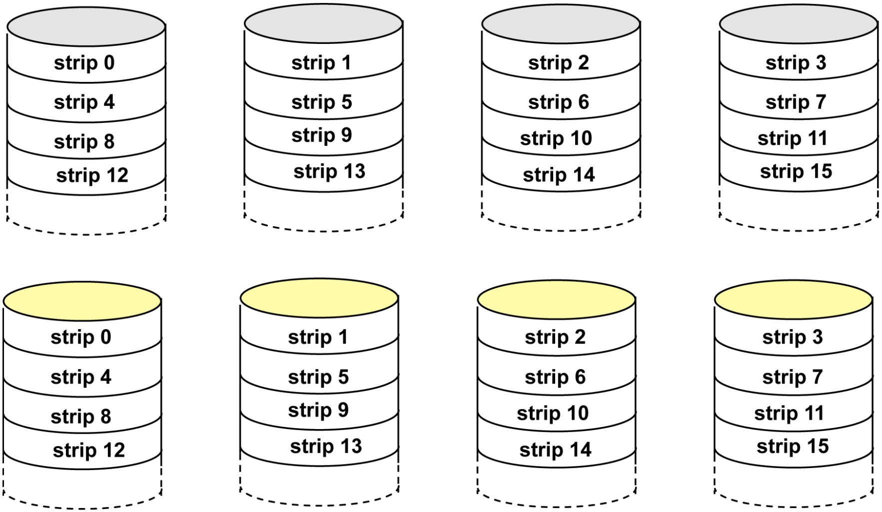
C
P
(a) RAID 0: non-redundant striping.
(b) RAID 1: mirrored disks.
(c) RAID 2: memory-style error-correcting codes.
(d) RAID 3: bit-interleaved parity.
(e) RAID 4: block-interleaved parity.
(f) RAID 5: block-interleaved distributed parity.
(g) RAID 6: P + Q redundancy.

RAID 0 (without any redundancy)

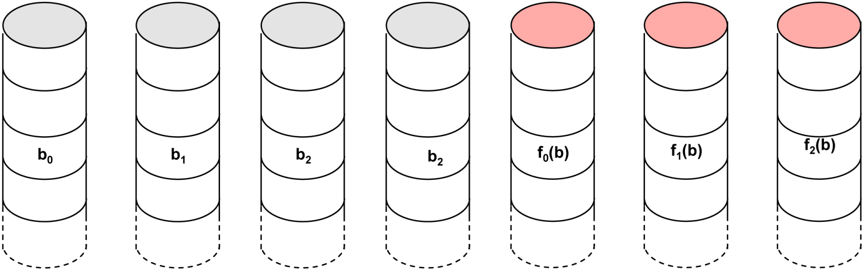


- **Decreased availability** compared to the **SLED**
- Increased bandwidth to/from logical disk

RAID 1 (mirrored)

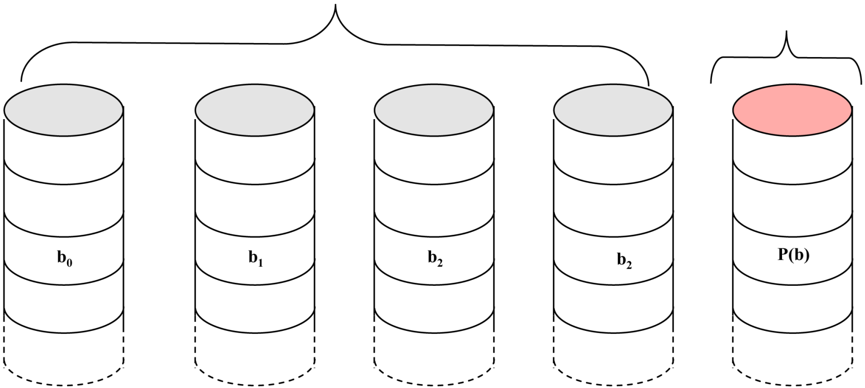


RAID 2 (redundancy through Hamming Code)



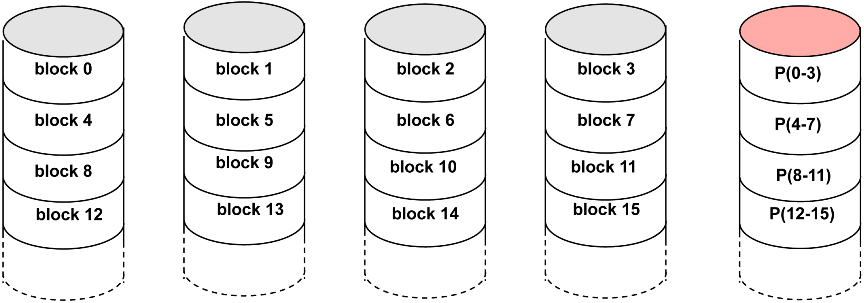
RAID 2 is an overkill and rarely implemented. Hamming code used for $f(b)$, b are very small strips, still a remarkable disk overhead compared to RAID 0.

RAID 3 (byte-interleaved parity)



- Disks spin in lockstep
- High throughput (e.g. for multimedia file systems)

RAID 4 (block-interleaved parity)

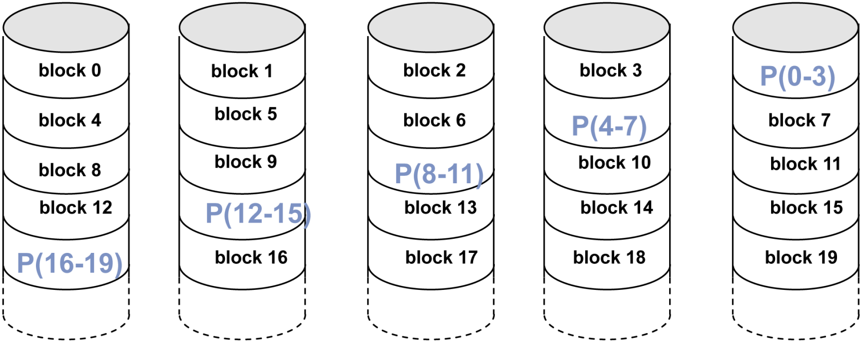


Parity computation: $P(0\ldots3) = block_0 \otimes block_1 \otimes block_2 \otimes block_3$

Result:

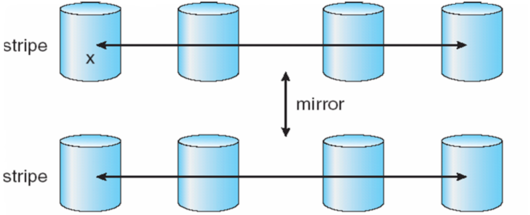
Small updates require 2 reads (old block + parity) and 2 writes (new block + parity) to update a single disk block. Parity disk may be a bottleneck.

RAID 5 (block-level distributed parity)

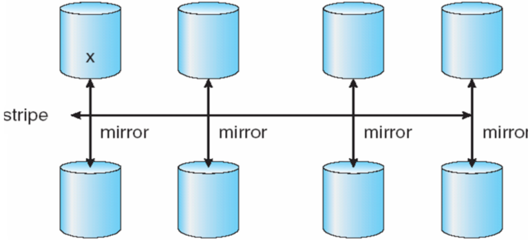


- Like RAID 4, but we distribute parity block on all disks ⇒ no longer a "bottleneck disk"
- Update performance still less than on a SLED
- Reconstruction after a failure is a bit tricky

RAID (0 + 1) and (1 + 0)



a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.

Tertiary Storage Devices

- Low cost is the defining characteristic of tertiary storage
- Generally, tertiary storage is built using *removable media*
- Two aspects of speed in tertiary storage are bandwidth and latency
- Bandwidth is measured in bytes per second
  - Sustained bandwidth – average data rate during a large transfer: $\frac{\text{\# of bytes}}{\text{transfer time}}$.
    Data rate when the data stream is actually flowing.
  - Effective bandwidth – average over the entire I/O time, including **seek** or **locate**, and cartridge switching. Drive's overall data rate.
- Common examples of removable media are removable disks, tapes and optical drives (e.g. DVDs)

Optical Disks (e.g. DVD, CD)

- The data on read-write disks can be modified over and over
  - To write a bit, a laser light heats up phase-change material and brings it to amorphous or crystalline state.
- WORM (“Write Once, Read Many Times”) disks can be written only once.
  - To write a bit, a laser light heats up an organic dye
  - Very durable and reliable.

Magnetic Tapes

- Kept in spool and wound or rewound past read-write head
  - Once data under head, transfer rates comparable to disk (160-360 MB/s)
  - 6.4 (LTO-7) - 10 TB (IBM 3532) typical capacity
  - Serpentine writng (e.g. 36 tracks x 80 passes)
  - Durability: 30 yrs, 16000 end to end passes
- Compared to a disk, a tape is less expensive and holds more data, but random access is much slower (up to 80 s).
- Tape is an economical medium for purposes that do not require fast random access, e.g. backup copies of disk data, holding huge volumes of data.
- Large tape installations typically use robotic tape changers that move tapes between tape drives and storage slots in a tape library.
  - stacker – library that holds a few tapes
  - silo – library that holds thousands of tapes
- A disk-resident file can be *archived* to tape for low cost storage; the computer can *stage* it back into disk storage for active use.

Application Interface

- Most OSs handle removable disks almost exactly like fixed disks – a new cartridge is formatted and an empty file system is generated on the disk.
- Tapes are presented as a raw storage medium, i.e. and an application does not open a file on the tape, it opens the whole tape drive as a raw device.
- Usually the tape drive is reserved for the exclusive use of that application.
- Since the OS does not provide file system services, the application must decide how to use the array of blocks.
- Since every application makes up its own rules for how to organize a tape, a tape full of data can generally only be used by the program that created it.

Tape Drives

- The basic operations for a tape drive differ from those of a disk drive.
- `locate` positions the tape to a specific logical block, not an entire track (corresponds to `seek`).
- The `read position` operation returns the logical block number where the tape head is.
- The `space` operation enables relative motion.
- Tape drives are “append-only” devices; updating a block in the middle of the tape also effectively erases everything beyond that block
- An EOT mark is placed after a block is written.

Hierarchical Storage Management (HSM)

- A hierarchical storage system extends the storage hierarchy beyond primary memory and secondary storage to incorporate tertiary storage – usually implemented as a jukebox of tapes or removable disks.
- Usually incorporate tertiary storage by extending the file system.
  - Small and frequently used files remain on disk.
  - Large, old, inactive files are archived to the jukebox.
- HSM is usually found in supercomputing centers and other large installations that have enormous volumes of data.