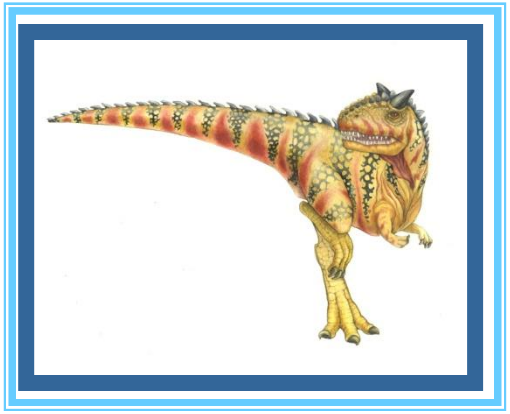
Chapter 12: Secondary-Storage Structure



Chapter 12: Secondary-Storage Structure

- Overview of Mass Storage Structure
- Disk Structure
- Disk Attachment
- Disk Scheduling
- Disk Management
- Swap-Space Management
- RAID Structure
- Disk Attachment
- Stable-Storage Implementation
- Tertiary Storage Devices
- Operating System Issues
- Performance Issues

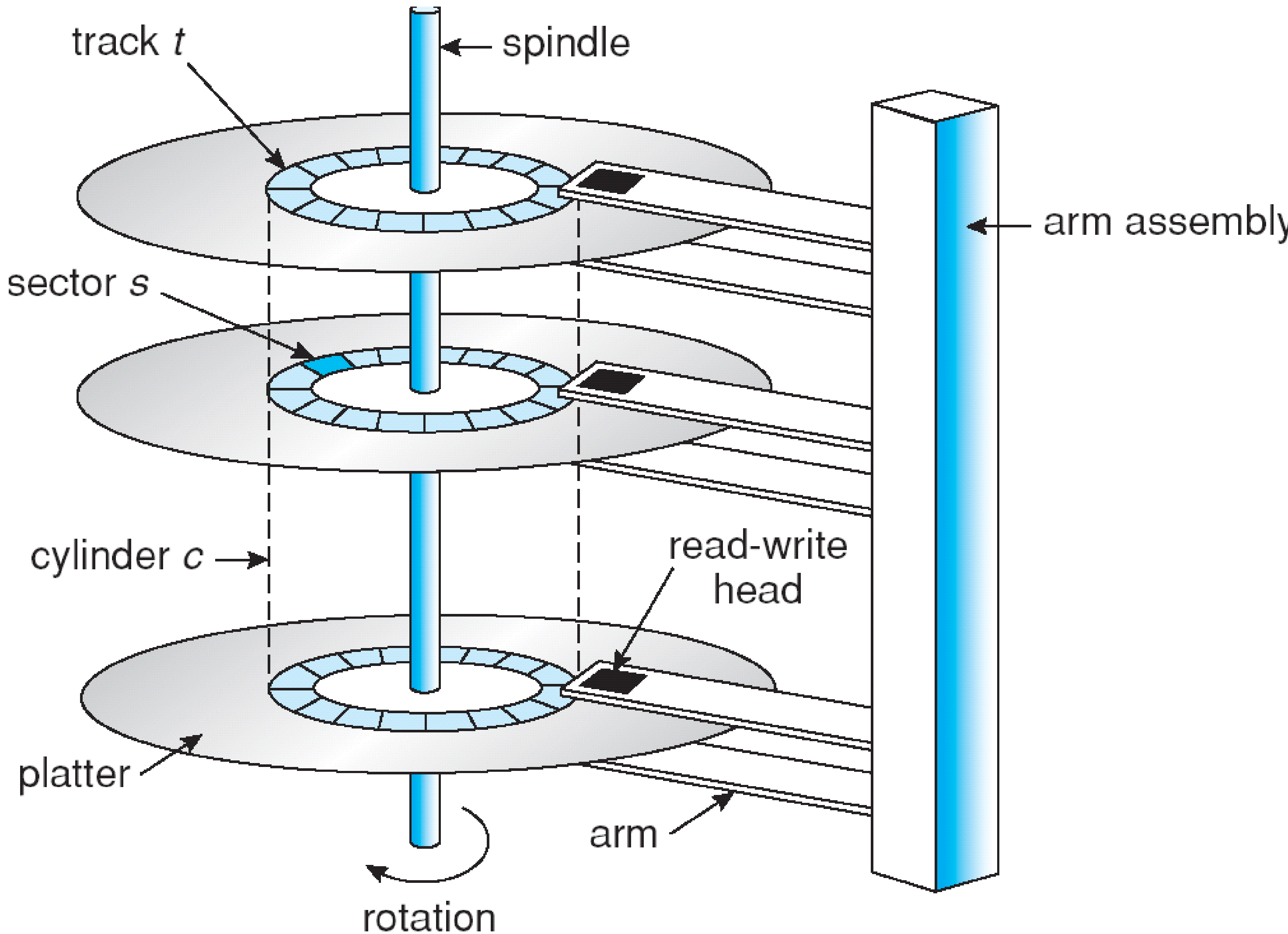
Objectives

- Describe the physical structure of secondary and tertiary storage devices and the resulting effects on the uses of the devices
- Explain the performance characteristics of mass-storage devices
- Discuss operating-system services provided for mass storage, including RAID and HSM

Overview of Mass Storage Structure

- Magnetic disks provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 200 times per second
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface
    - That's bad
- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE, ATA, SATA, USB, Fibre Channel, SCSI**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

Moving-head Disk Mechanism



Overview of Mass Storage Structure (Cont.)

- Magnetic tape
  - Was early secondary-storage medium
  - Relatively permanent and holds large quantities of data
  - Access time slow
  - Random access ~1000 times slower than disk
  - Mainly used for backup, storage of infrequently-used data, transfer medium between systems
  - Kept in spool and wound or rewound past read-write head
  - Once data under head, transfer rates comparable to disk
  - 20-200GB typical storage
  - Common technologies are 4mm, 8mm, 19mm, LTO-2 and SDLT

Disk Structure

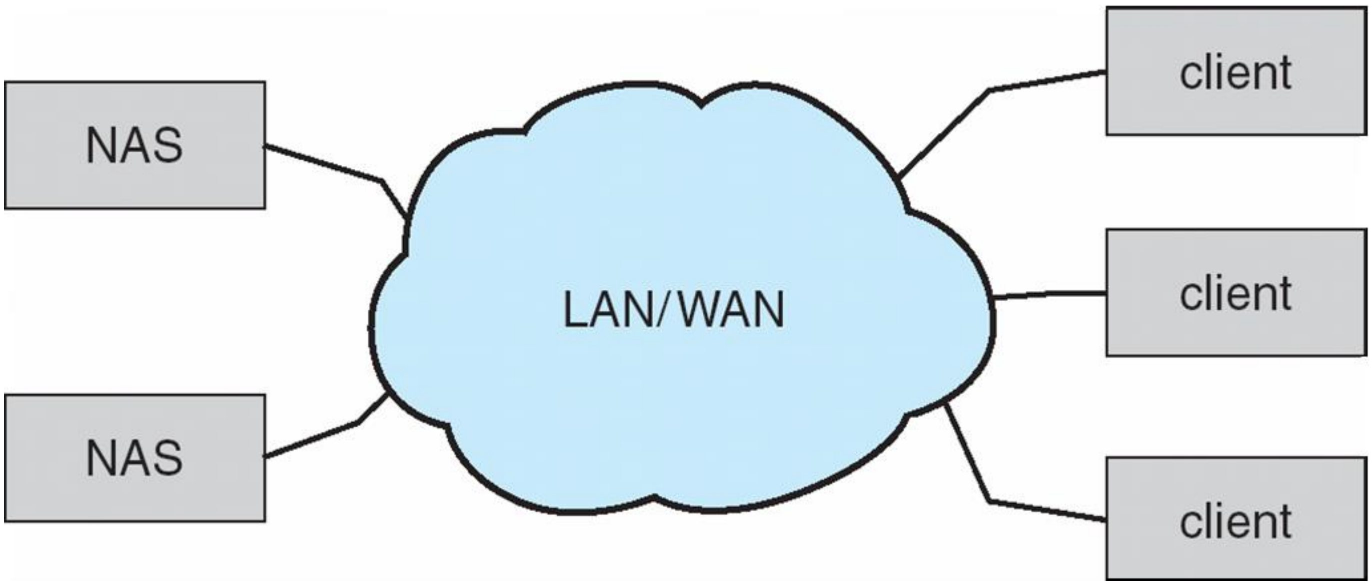
- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

Disk Attachment

- Host-attached storage accessed through I/O ports talking to I/O busses
- SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** requests operation and **SCSI targets** perform tasks
  - Each target can have up to 8 **logical units** (disks attached to device controller
- FC is high-speed serial architecture
  - Can be switched fabric with 24-bit address space – the basis of **storage area networks** (**SAN**s) in which many hosts attach to many storage units
  - Can be **arbitrated loop** (**FC-AL**) of 126 devices

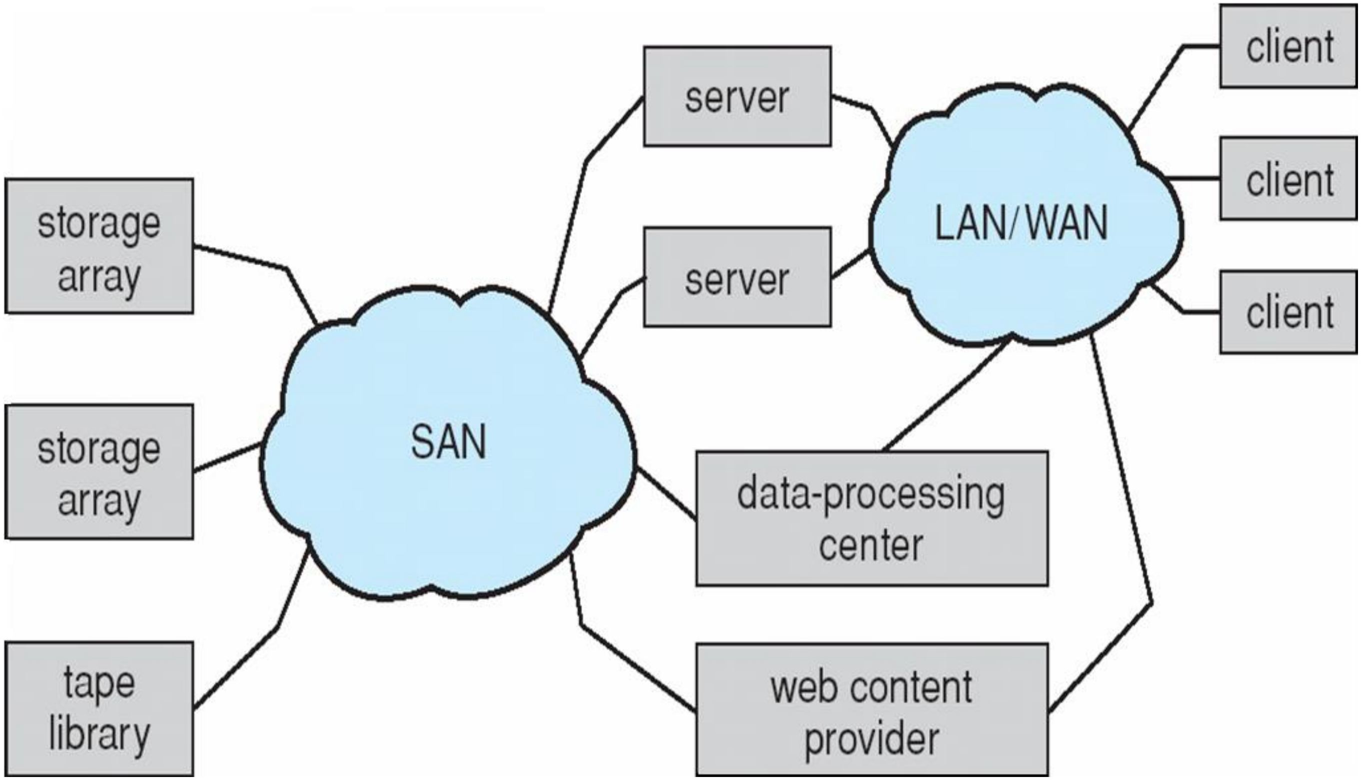
Network-Attached Storage

- Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)
- NFS and CIFS are common protocols
- Implemented via remote procedure calls (RPCs) between host and storage
- New iSCSI protocol uses IP network to carry the SCSI protocol



Storage Area Network

- Common in large storage environments (and becoming more common)
- Multiple hosts attached to multiple storage arrays - flexible



Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth.
- Access time has two major components
  - *Seek time* is the time for the disk are to move the heads to the cylinder containing the desired sector.
  - *Rotational latency* is the additional time waiting for the disk to rotate the desired sector to the disk head.
- Minimize seek time
- Seek time $\approx$ seek distance
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

Disk Scheduling (Cont.)

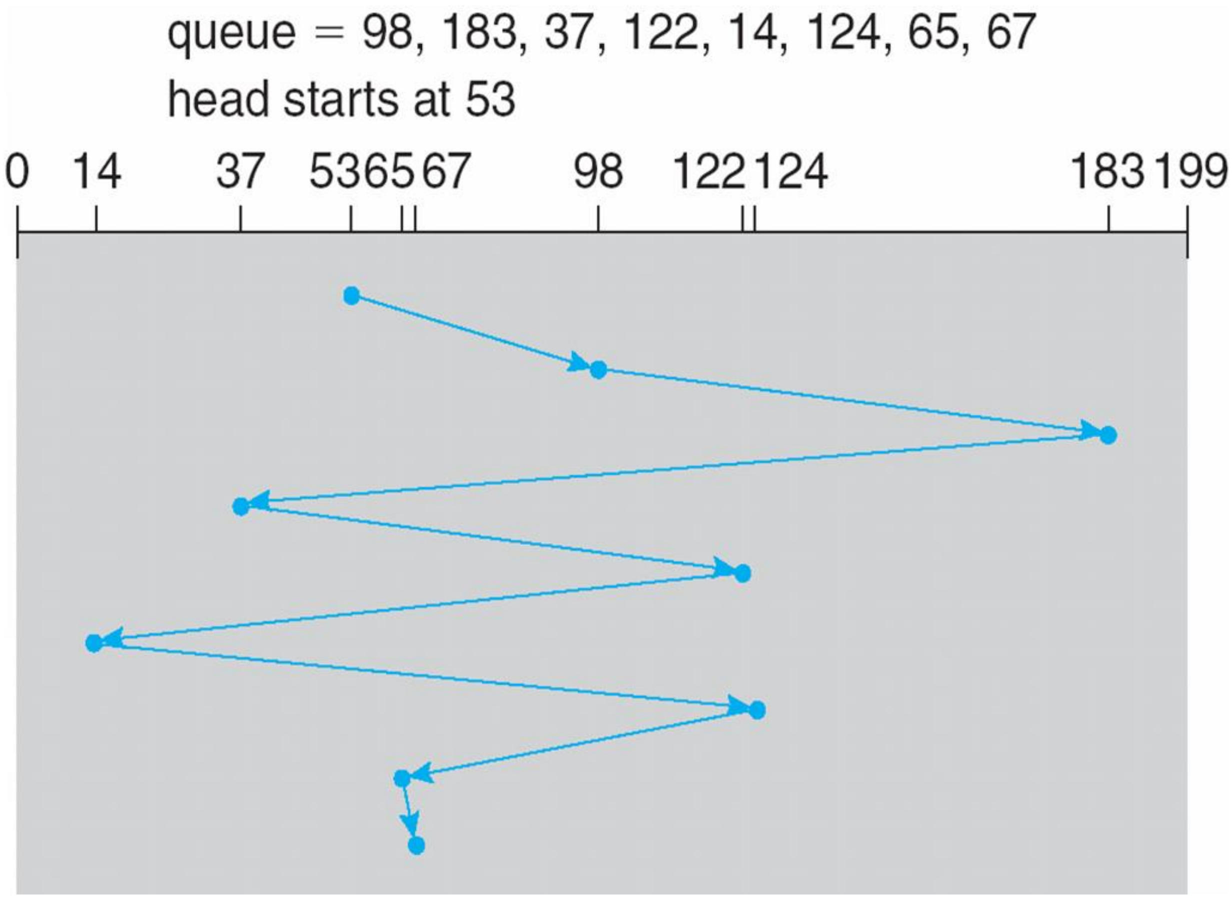
- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

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

Head pointer 53

FCFS

Illustration shows total head movement of 640 cylinders.



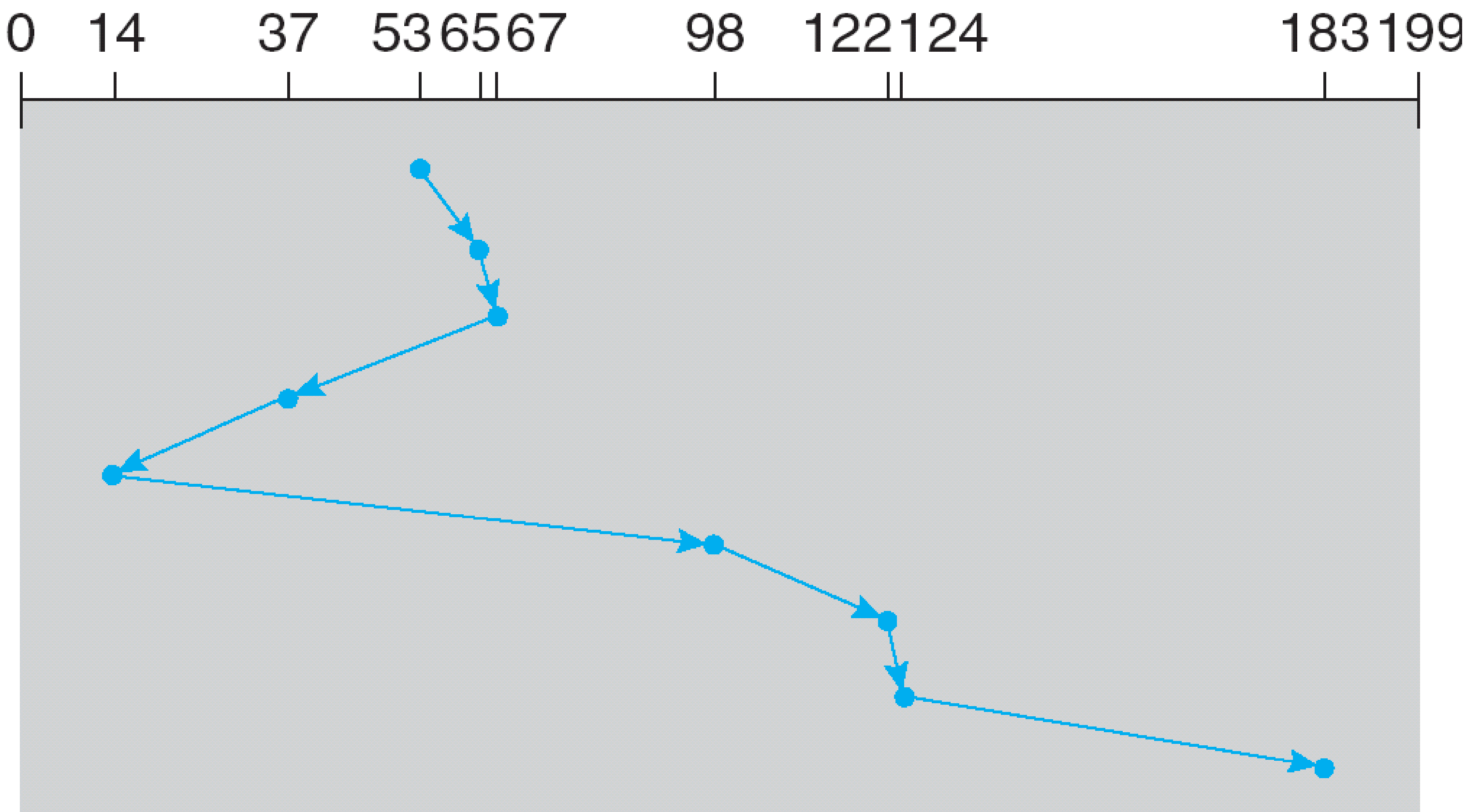
SSTF

- Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.
- Illustration shows total head movement of 236 cylinders.

SSTF (Cont.)

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

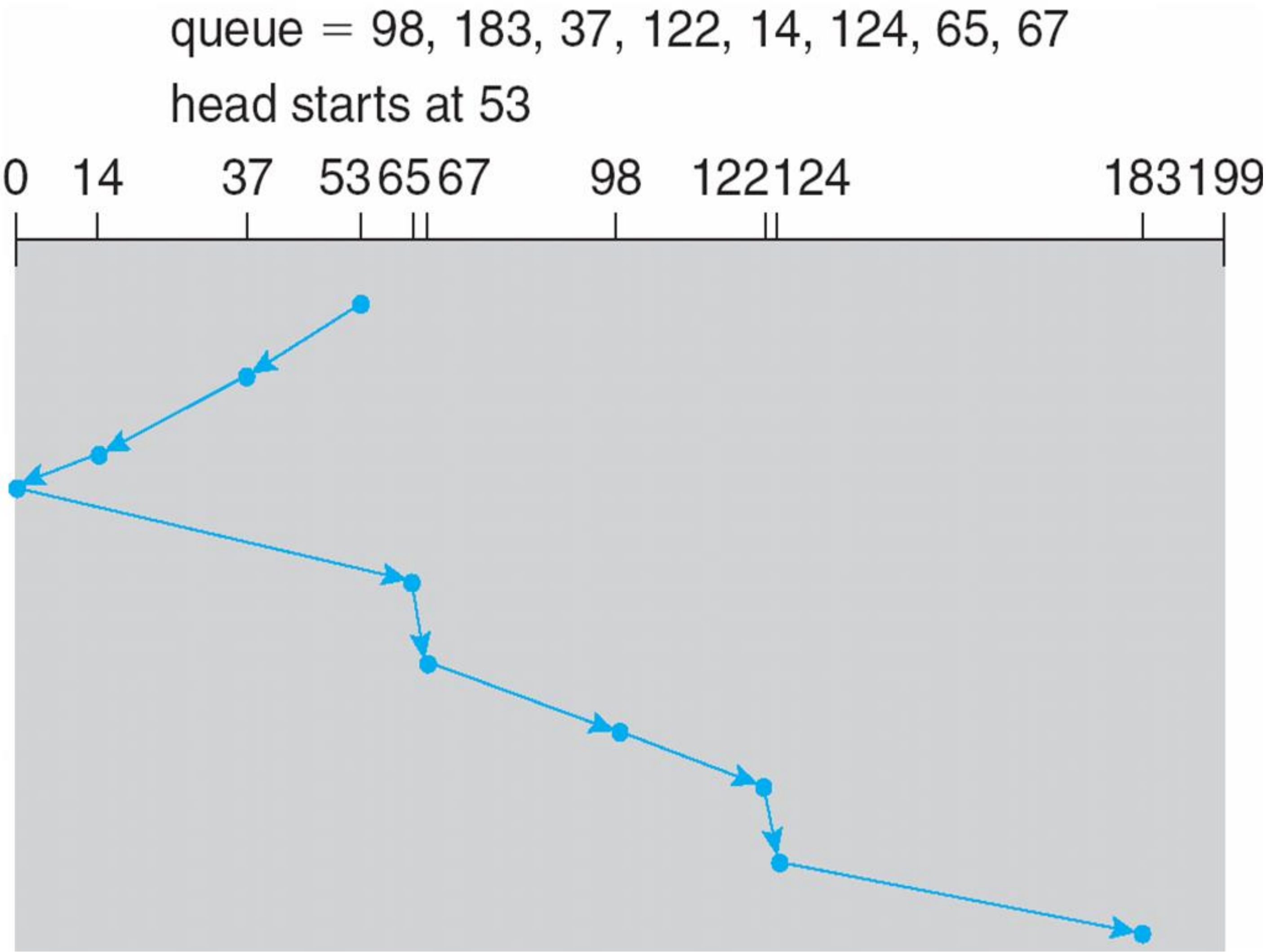
head starts at 53



SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm*.
- Illustration shows total head movement of 208 cylinders.

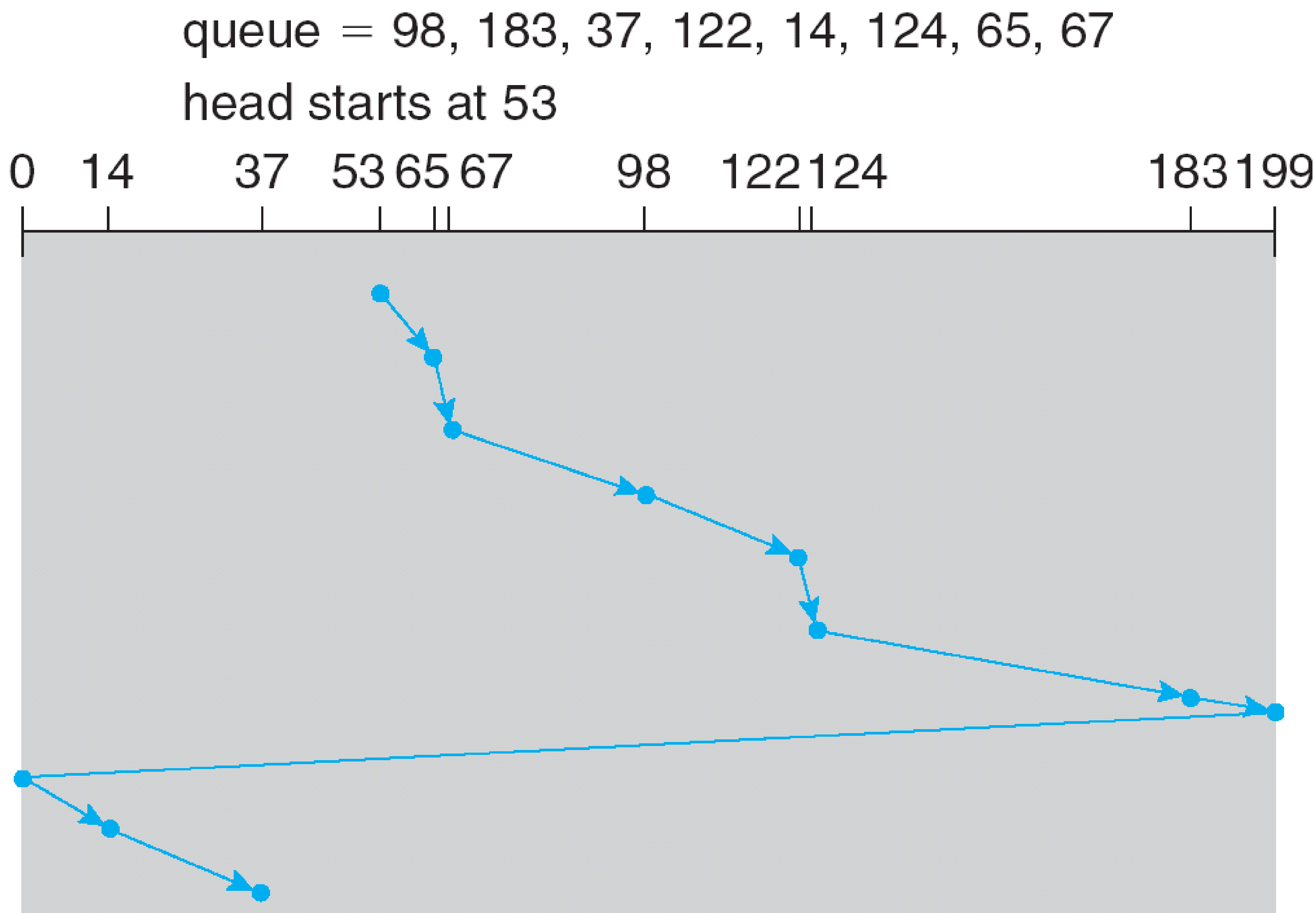
SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-SCAN

- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

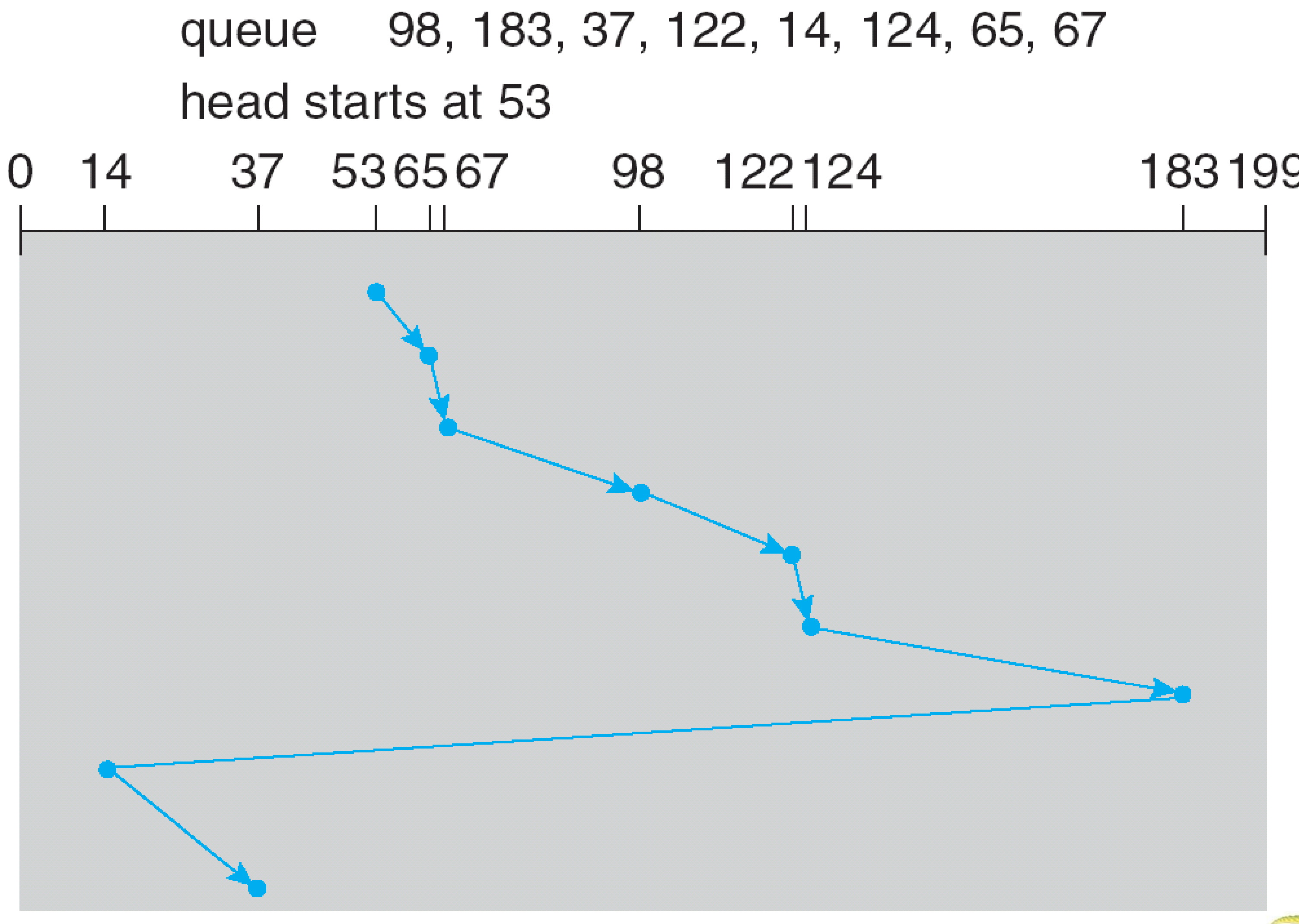
C-SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0 14 37 53 65 67 98 122 124 183 199

C-LOOK

- Version of C-SCAN
- 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.

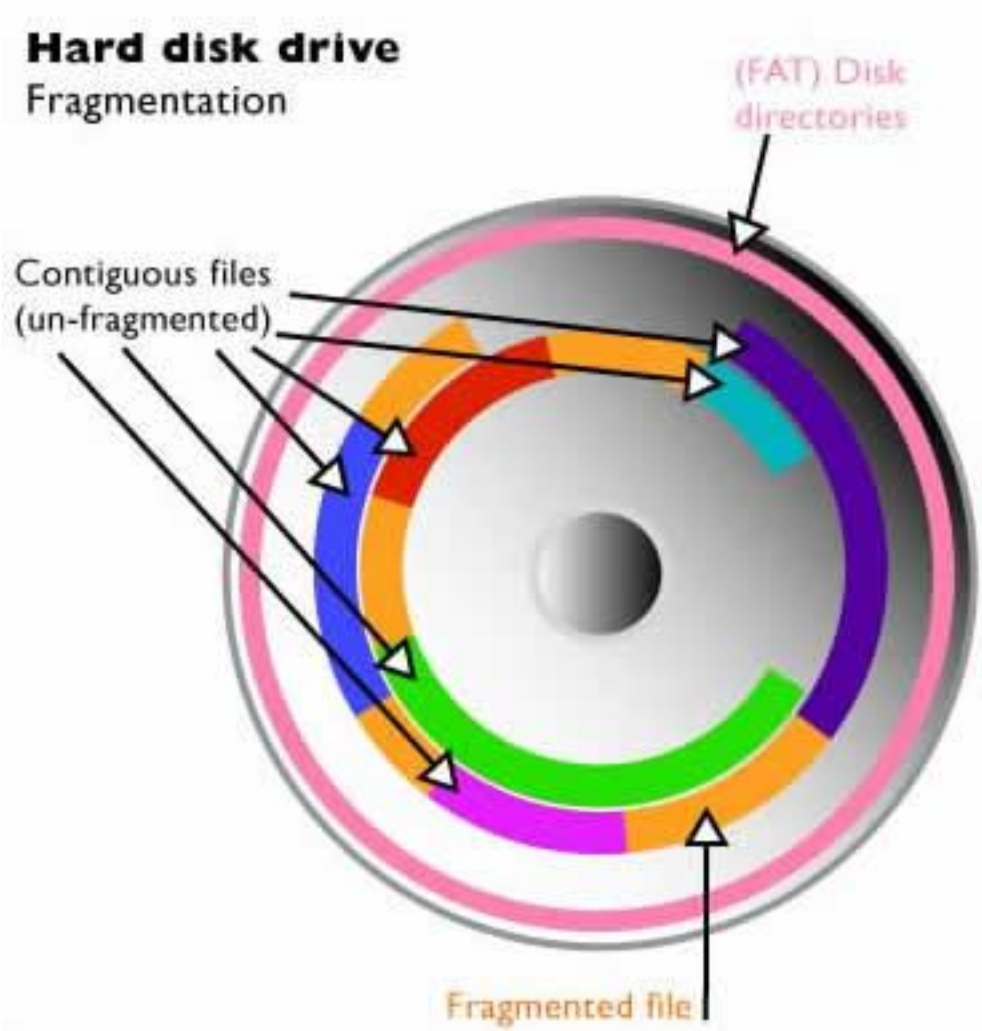
C-LOOK (Cont.)


queue 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

Selecting a Disk-Scheduling Algorithm

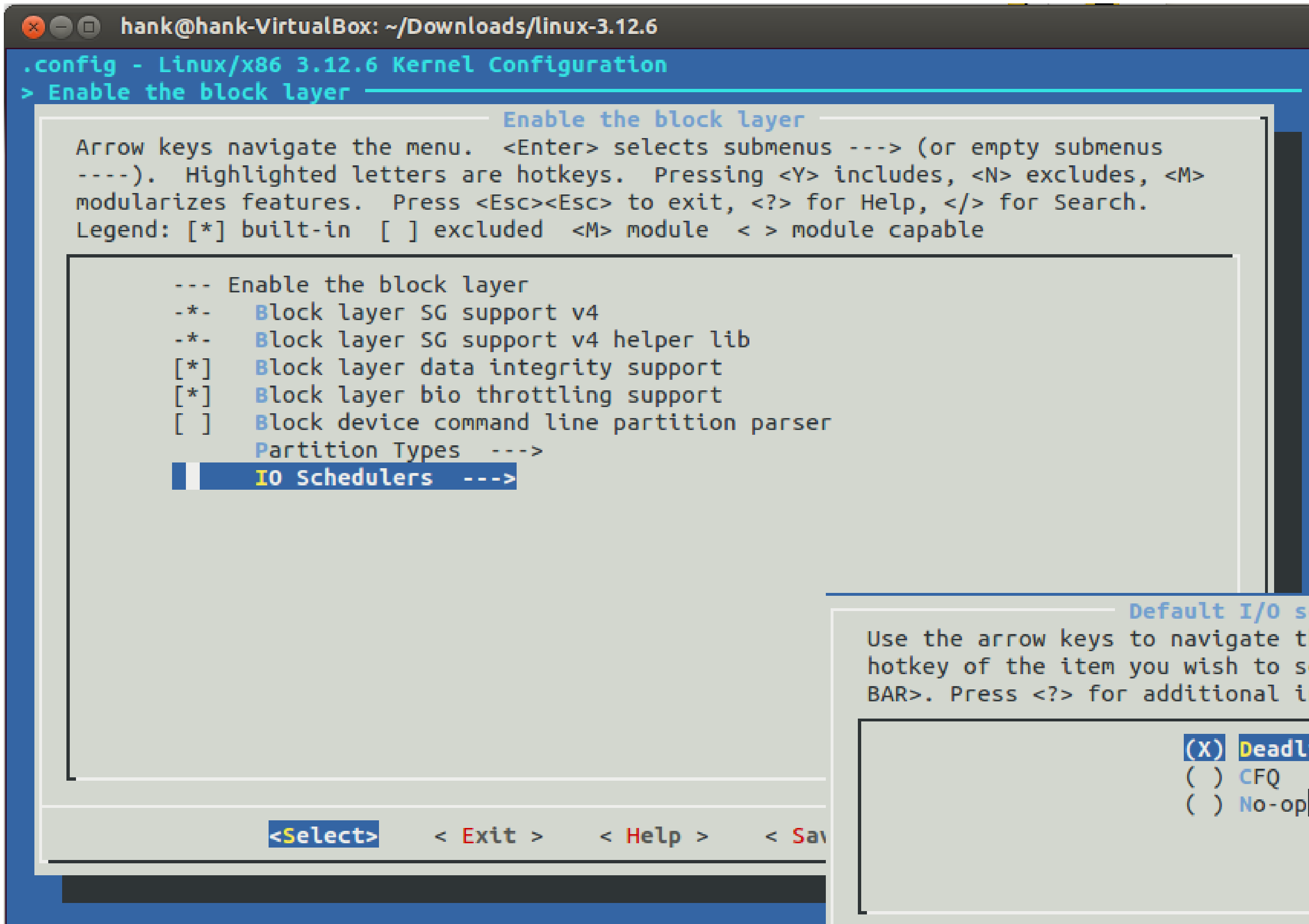
- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

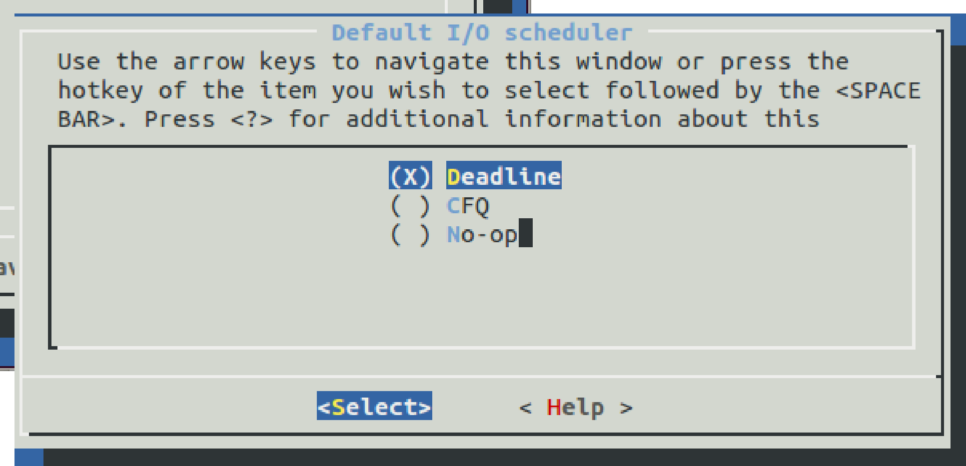
Selecting a Disk-Scheduling Algorithm



Selecting a Disk Scheduling Algorithm

- http://www.gnutoolbox.com/linux-io-elevator/





Selecting a Disk Scheduling Algorithm

linux1.cs.nctu.edu.tw - PuTTY

```
linux1 [/sys/block/sda/queue] -ysw- % ls
add_random           iostats                  max_segment_size     read_ahead_kb
discard_granularity  logical_block_size       minimum_io_size      rotational
discard_max_bytes    max_hw_sectors_kb        nomerges             rq_affinity
discard_zeroes_data  max_integrity_segments   nr_requests          scheduler
hw_sector_size       max_sectors_kb           optimal_io_size      write_same_max_bytes
iosched              max_segments             physical_block_size
linux1 [/sys/block/sda/queue] -ysw- %
linux1 [/sys/block/sda/queue] -ysw- % cat scheduler
noop deadline [cfq]
linux1 [/sys/block/sda/queue] -ysw- %
```

TCQ and NCQ

- http://wdc.custhelp.com/app/answers/detail/a_id/1311/~/support-for-ncq-and%2For-tcq-on-wd-sata-3gb%2Fs-%28sata-ii%29-drives

Definition of Native Command Queuing (NCQ)

- Native Command Queuing allows multiple commands to be outstanding within a drive at the same time. Hard drives that support NCQ have an internal queue where outstanding commands can be dynamically rescheduled or re-ordered, along with the necessary tracking mechanisms for outstanding and completed portions of the workload. NCQ also has a mechanism that allows the host to issue additional commands to the drive while the drive is seeking data for another command.

- NCQ allows the drive to set up the direct memory access (DMA) operation for a data transfer without host software intervention. This is also called first-party DMA – it means that the device is capable of complex sequences of operations without CPU intervention. The drive itself knows the current angular and rotational position of the drive head. The drive then selects the next data transfer to minimize both seek and rotational latencies.

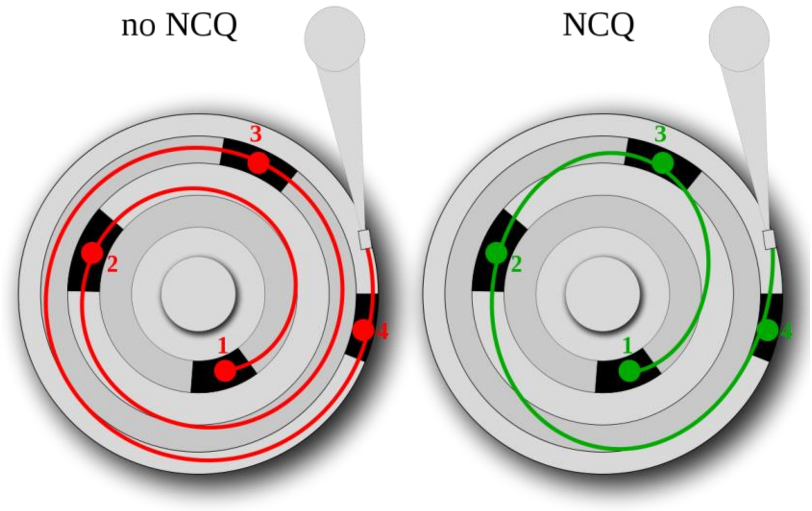
Definition of Tagged Command Queuing (TCQ)

- Tagged Command Queuing is a standard as a way to allow hard drives to accept multiple concurrent commands from a host PC. When commands arrive at the drive's buffer, they are tagged with an identifier and then reordered by the drive's microprocessor to minimize the distance the drive's read head needs to move laterally along the platter looking for data. For example, if a command is looking for data in one section of the drive and a following queued command is looking for data in a neighboring area, the host adapter can reorder the commands to make the two occur sequentially. This is a different system than IDE/ATA, which will allow only a single command to be outstanding at a time to any device and processes requests serially.

All other Western Digital Serial ATA hard drives (WD360GD, WDxxxxJD, WDxxxxKD, and WDxxxxSD) do not support any Queuing method. TCQ and NCQ are slightly different variations of Queuing and are not interchangeable. A hard drive that supports TCQ will not perform Queuing when attached to a controller that support NCQ (and vice versa). For a detailed explanation of TCQ please visit http://www.wdc.com/en/library/sata/2579-001076.pdf

TCQ and NCQ

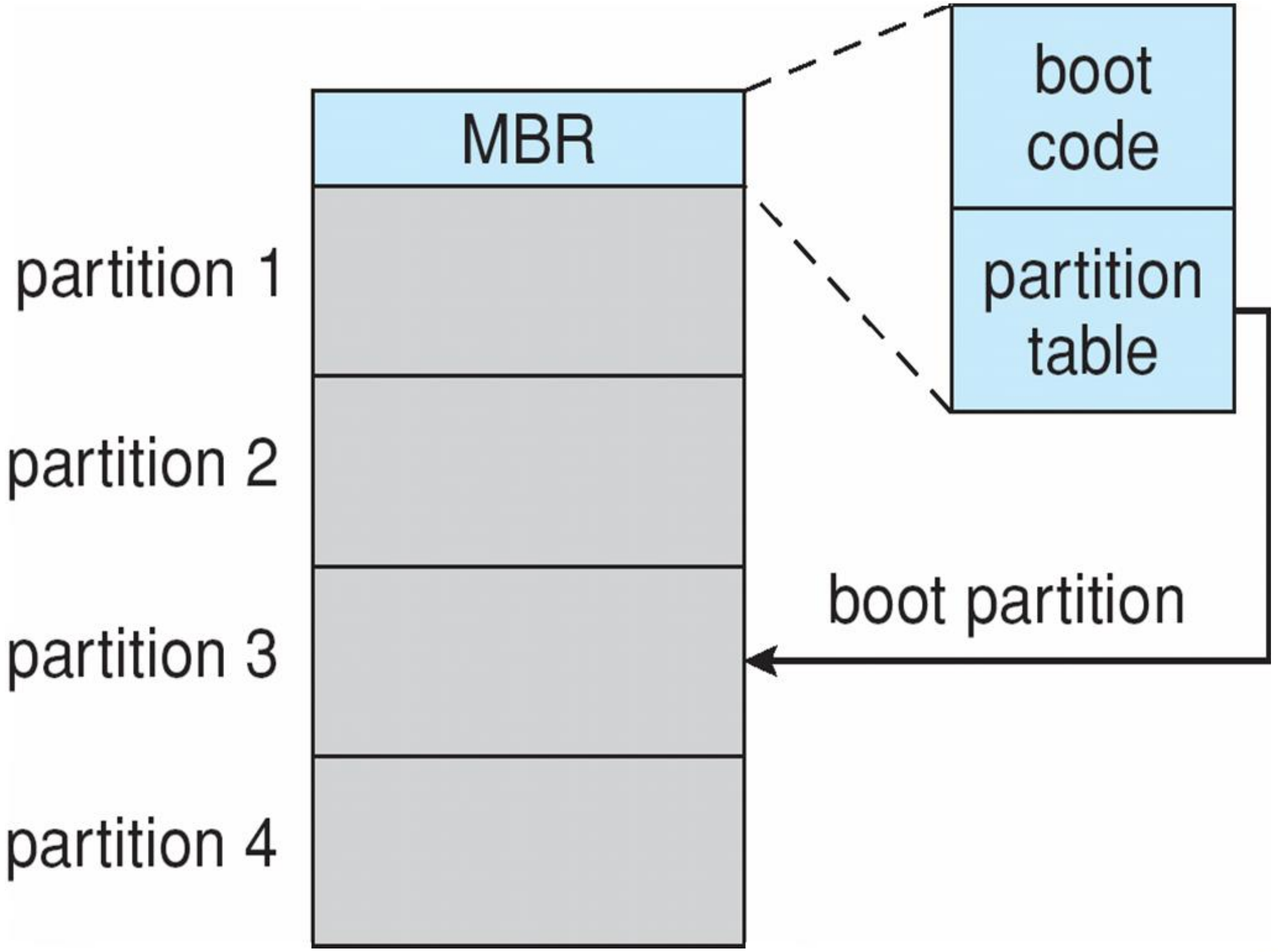
- http://en.wikipedia.org/wiki/Native_Command_Queuing



Disk Management

- *Low-level formatting*, or *physical formatting* — Dividing a disk into sectors that the disk controller can read and write.
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk.
  - *Partition* the disk into one or more groups of cylinders.
  - *Logical formatting* or “making a file system”.
- Boot block initializes system.
  - The bootstrap is stored in ROM.
  - *Bootstrap loader* program.
- Methods such as *sector sparing* used to handle bad blocks.

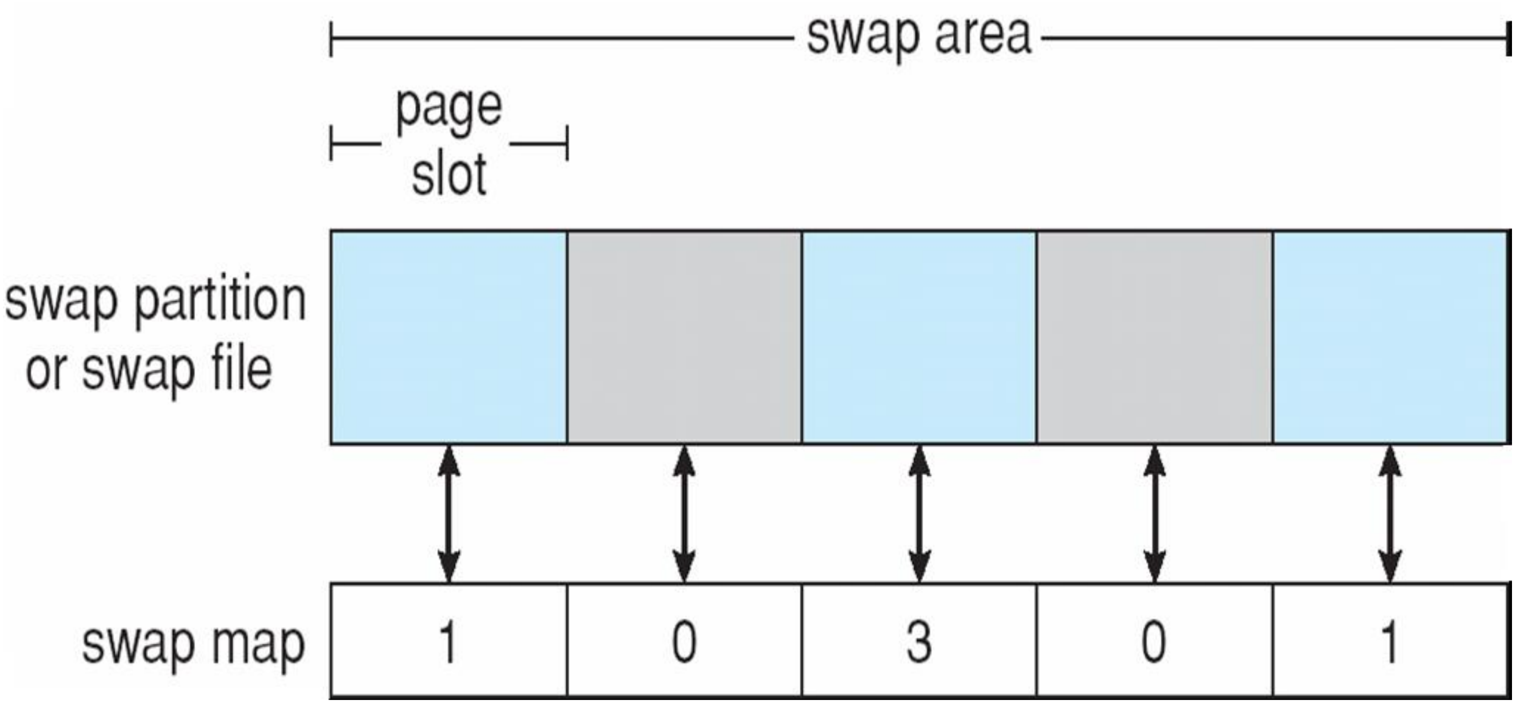
Booting from a Disk in Windows 2000



Swap-Space Management

- Swap-space — Virtual memory uses disk space as an extension of main memory.
- Swap-space can be carved out of the normal file system,or, more commonly, it can be in a separate disk partition.
- Swap-space management
  - 4.3BSD allocates swap space when process starts; holds *text segment* (the program) and *data segment.*
  - Kernel uses *swap maps* to track swap-space use.
  - Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created.

Data Structures for Swapping on Linux Systems



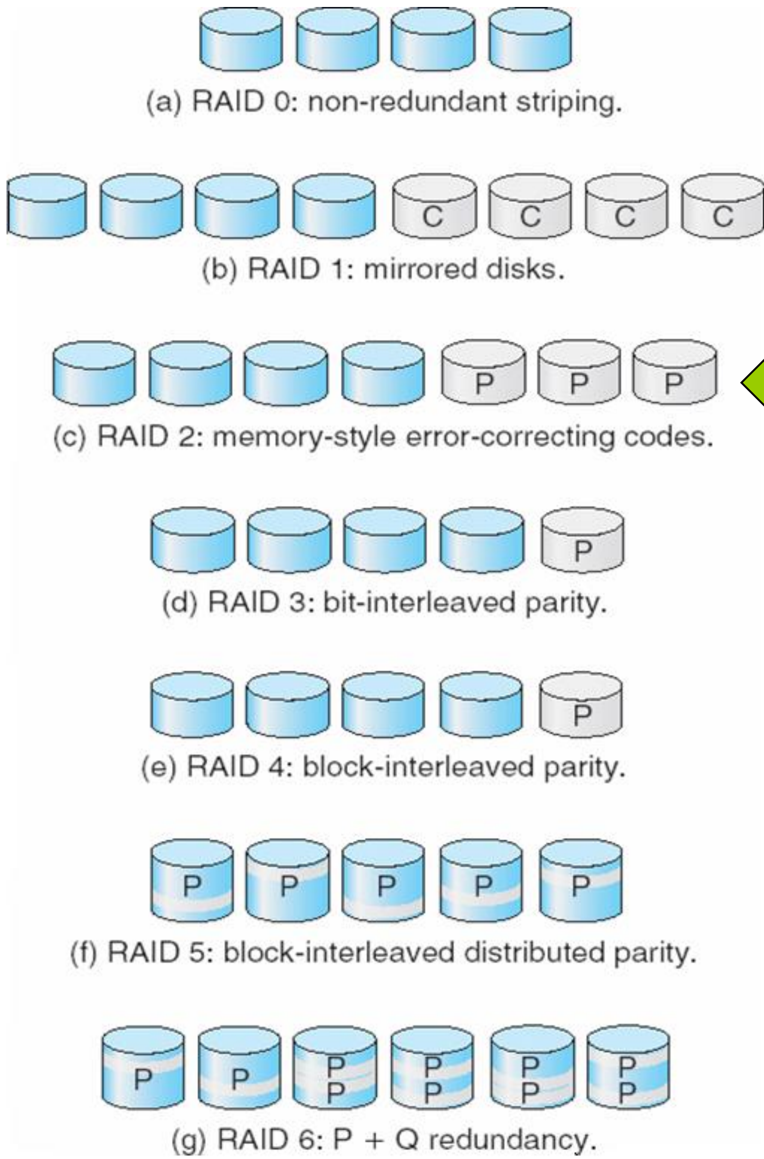
RAID Structure

- **RAID** – multiple disk drives provides **reliability** via **redundancy**.
- RAID is arranged into six different levels.

RAID (cont)

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.
- Disk striping uses a group of disks as one storage unit.
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
  - *Mirroring* or *shadowing* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy.

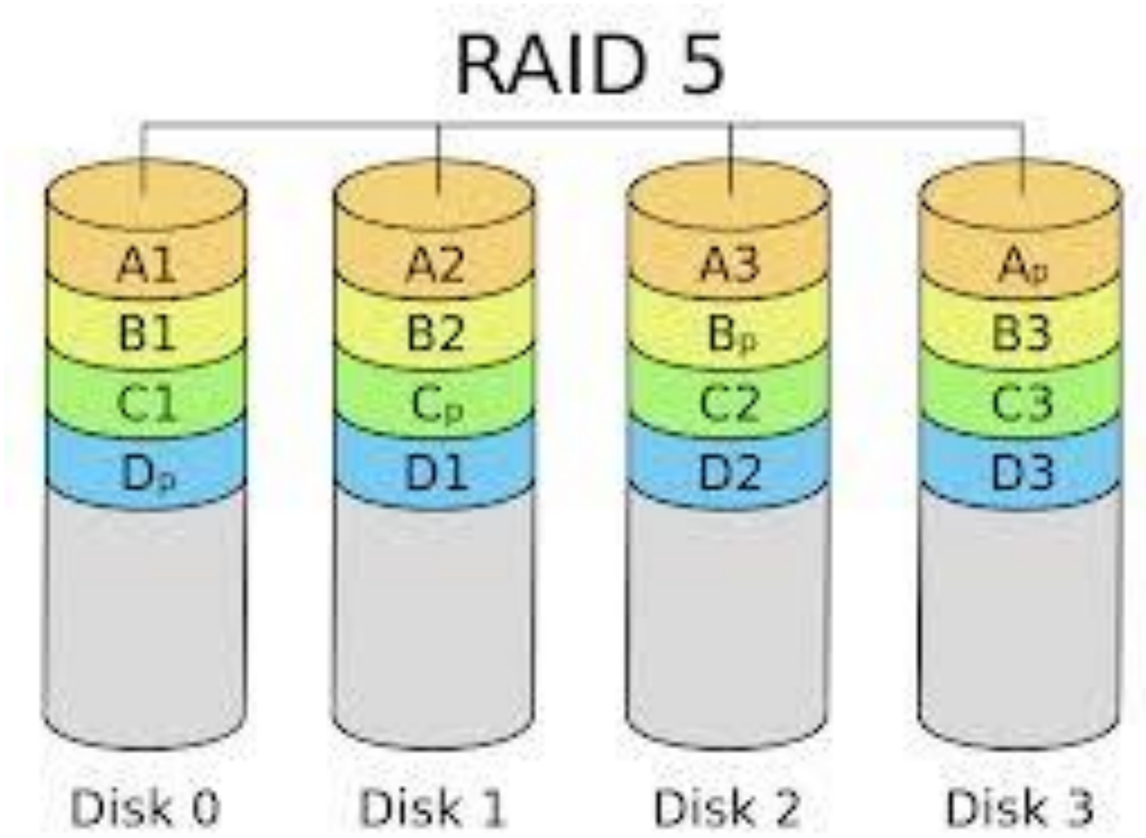
RAID Levels



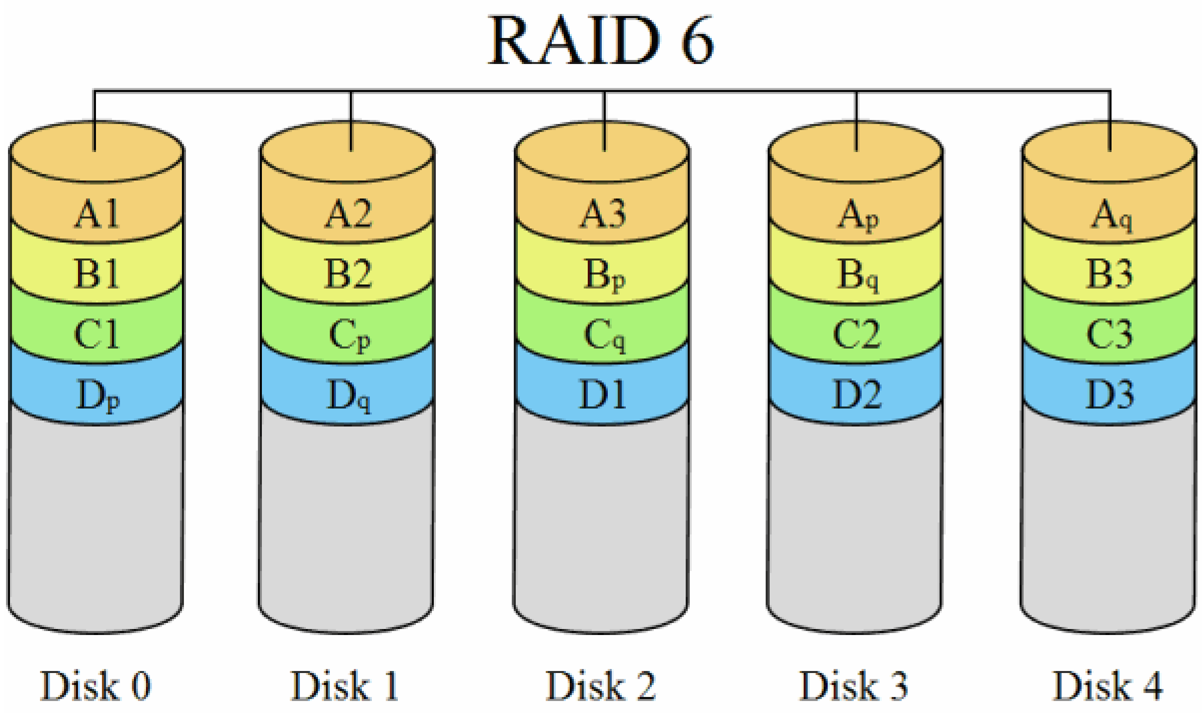
Hamming code(7,4)

http://en.wikipedia.org/wiki/RAID

Raid 5

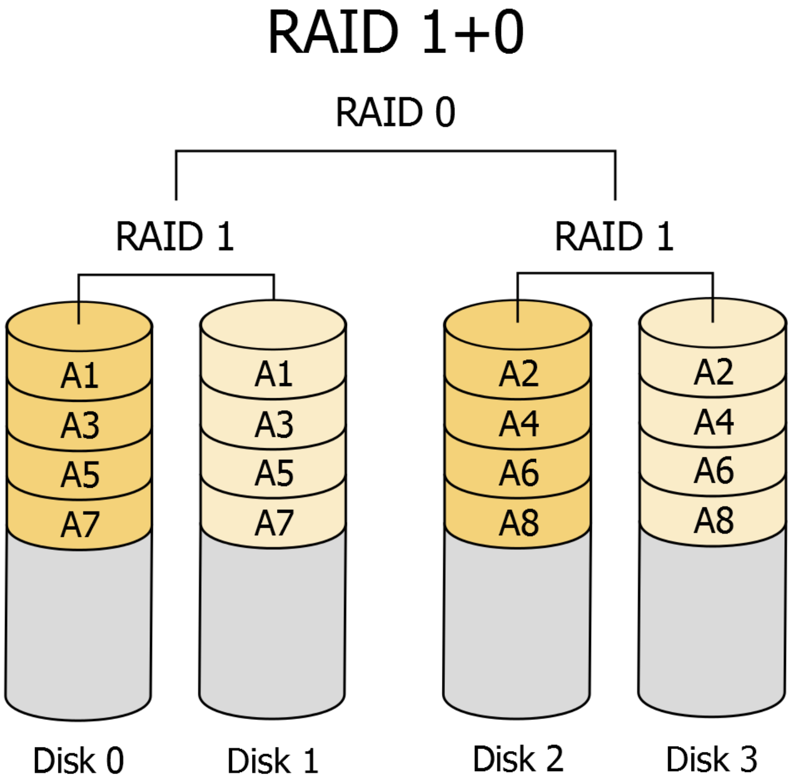

RAID 5
A1
A2
A3
Ap
B1
B2
Bp
B3
C1
Cp
C2
C3
Dp
D1
D2
D3
Disk 0
Disk 1
Disk 2
Disk 3

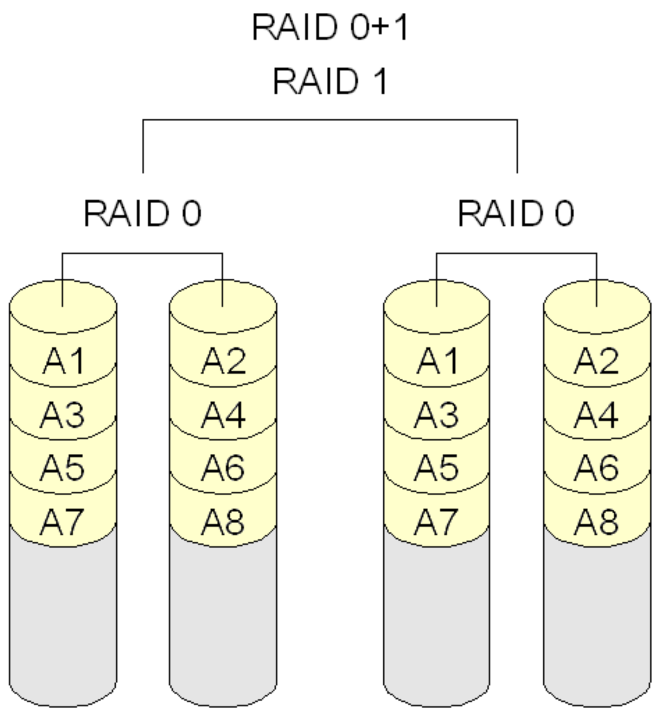
Raid 6



RAID 6
A1
B1
C1
Dp
A2
B2
Cp
Dq
A3
Bp
Cq
D1
Ap
Bq
C2
D2
Aq
B3
C3
D3
Disk 0
Disk 1
Disk 2
Disk 3
Disk 4

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





Stable-Storage Implementation

- Write-ahead log scheme requires stable storage.
- To implement stable storage:
  - Replicate information on more than one nonvolatile storage media with independent failure modes.
  - Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery.

Tertiary Storage Devices

- Low cost is the defining characteristic of tertiary storage.
- Generally, tertiary storage is built using *removable media*
- Common examples of removable media are floppy disks and CD-ROMs; other types are available.

Removable Disks

- Floppy disk — thin flexible disk coated with magnetic material, enclosed in a protective plastic case.
  - Most floppies hold about 1 MB; similar technology is used for removable disks that hold more than 1 GB.
  - Removable magnetic disks can be nearly as fast as hard disks, but they are at a greater risk of damage from exposure.

Removable Disks (Cont.)

- A magneto-optic disk records data on a rigid platter coated with magnetic material.
  - Laser heat is used to amplify a large, weak magnetic field to record a bit.
  - Laser light is also used to read data (Kerr effect).
  - The magneto-optic head flies much farther from the disk surface than a magnetic disk head, and the magnetic material is covered with a protective layer of plastic or glass; resistant to head crashes.
- Optical disks do not use magnetism; they employ special materials that are altered by laser light.

WORM Disks

- The data on read-write disks can be modified over and over.
- WORM ("Write Once, Read Many Times") disks can be written only once.
- Thin aluminum film sandwiched between two glass or plastic platters.
- To write a bit, the drive uses a laser light to burn a small hole through the aluminum; information can be destroyed by not altered.
- Very durable and reliable.
- *Read Only* disks, such ad CD-ROM and DVD, come from the factory with the data pre-recorded.

Tapes

- Compared to a disk, a tape is less expensive and holds more data, but random access is much slower.
- 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.

Magnetic tape to the rescue

Information storage: A 60-year-old technology offers a solution to a modern problem—how to store all those bits and bytes cheaply and reliably

Nov 30th 2013 | From the print edition

Like 1.2k Tweet 200

WHEN physicists switch on the Large Hadron Collider (LHC), between three and six gigabytes of data spew out of it every second. That is, admittedly, an extreme example. But the flow of data from smaller sources than CERN, the European particle-research organisation outside Geneva that runs the LHC, is also growing inexorably. At the moment it is doubling every two years. These data need to be stored. The need for mass storage is reviving a technology which, only a few years ago, seemed destined for the scrapheap: magnetic tape.

http://0rz.tw/7XMFw

Operating System Issues

- Major OS jobs are to manage physical devices and to present a virtual machine abstraction to applications

- For hard disks, the OS provides two abstraction:
  - Raw device – an array of data blocks.
  - File system – the OS queues and schedules the interleaved requests from several applications.

Raw Device I/O

```
DeviceIoControl (dev,
    FSCTL_ALLOW_EXTENDED_DASD_IO,
    NULL,
    0,
    NULL,
    0,
    &dwResult,
    NULL);


// If DASD is needed but we failed to obtain it, perform open - 'quick format' - close - open
// so that the filesystem driver does not prevent us from formatting hidden sectors.
for (nPass = (bFailedRequiredDASD ? 0 : 1); nPass < 2; nPass++)
{
    int retryCount;

    retryCount = 0;

    // Try exclusive access mode first
    // Note that when exclusive access is denied, it is worth retrying (usually succeeds after a few tries).
    while (dev == INVALID_HANDLE_VALUE && retryCount++ < EXCL_ACCESS_MAX_AUTO_RETRIES)
    {
        dev = CreateFile (devName, GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);

        if (retryCount > 1)
            Sleep (EXCL_ACCESS_AUTO_RETRY_DELAY);
    }

    if (dev == INVALID_HANDLE_VALUE)
    {
        // Exclusive access denied -- retry in shared mode
        dev = CreateFile (devName, GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, NULL);
        if (dev != INVALID_HANDLE_VALUE)
        {
            if (IDNO == MessageBoxW (volParams->hwndDlg, GetString ("DEVICE_IN_USE_FORMAT"), lpszTitle, MB_YESNO|MB_ICONWARNING|MB_DEFBUTTON2)
            {
                nStatus = ERR_DONT_REPORT;
                goto error;
            }
```

TrueCrypt 7.1a Source\Common\Format.c

Raw Device I/O

```
DeviceIoControl (dev,
    FSCTL_ALLOW_EXTENDED_DASD_IO,
    NULL,
    0,
    NULL,
    0,
    &dwResult,
    NULL);


// If DASD is needed but we failed to obtain it, perform open - 'quick format' - close - open
// so that the filesystem driver does not prevent us from formatting hidden sectors.
for (nPass = (bFailedRequiredDASD ? 0 : 1); nPass < 2; nPass++)
{
    int retryCount;

    retryCount = 0;

    // Try exclusive access mode first
    // Note that when exclusive access is denied, it is worth retrying (usually succeeds after a few tries).
    while (dev == INVALID_HANDLE_VALUE && retryCount++ < EXCL_ACCESS_MAX_AUTO_RETRIES)
    {
        dev = CreateFile (devName, GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, 0, NULL);

        if (retryCount > 1)
            Sleep (EXCL_ACCESS_AUTO_RETRY_DELAY);
    }

    if (dev == INVALID_HANDLE_VALUE)
    {
        // Exclusive access denied -- retry in shared mode
        dev = CreateFile (devName, GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, NULL);
        if (dev != INVALID_HANDLE_VALUE)
        {
            if (IDNO == MessageBoxW (volParams->hwndDlg, GetString ("DEVICE_IN_USE_FORMAT"), lpszTitle, MB_YESNO|MB_ICONWARNING|MB_DEFBUTTON2)
            {
                nStatus = ERR_DONT_REPORT;
                goto error;
            }
```

TrueCrypt 7.1a Source\Common\Format.c

Raw Device I/O

```c
            }
        }

        if (volParams->hiddenVol || bInstantRetryOtherFilesys)
            break;  // The following "quick format" operation would damage the outer volume

        if (nPass == 0)
        {
            char buf [2 * TC_MAX_VOLUME_SECTOR_SIZE];
            DWORD bw;

            // Perform pseudo "quick format" so that the filesystem driver does not prevent us from
            // formatting hidden sectors
            memset (buf, 0, sizeof (buf));

            if (!WriteFile (dev, buf, sizeof (buf), &bw, NULL))
            {
                nStatus = ERR_OS_ERROR;
                goto error;
            }

            FlushFileBuffers (dev);
            CloseHandle (dev);
            dev = INVALID_HANDLE_VALUE;
        }
    }

    if (DeviceIoControl (dev, FSCTL_IS_VOLUME_MOUNTED, NULL, 0, NULL, 0, &dwResult, NULL))
    {
        Error ("FORMAT_CANT_DISMOUNT_FILESYS");
        nStatus = ERR_DONT_REPORT;
        goto error;
    }
}
```

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 application does not 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 that is written.

File Naming

- The issue of naming files on removable media is especially difficult when we want to write data on a removable cartridge on one computer, and then use the cartridge in another computer.
- Contemporary OSs generally leave the name space problem unsolved for removable media, and depend on applications and users to figure out how to access and interpret the data.
- Some kinds of removable media (e.g., CDs) are so well standardized that all computers use them the same way.

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.

Speed

- 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; # of bytes/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.

Speed (Cont.)

- Access latency – amount of time needed to locate data.
  - Access time for a disk – move the arm to the selected cylinder and wait for the rotational latency; < 35 milliseconds.
  - Access on tape requires winding the tape reels until the selected block reaches the tape head; tens or hundreds of seconds.
  - Generally say that random access within a tape cartridge is about a thousand times slower than random access on disk.
- The low cost of tertiary storage is a result of having many cheap cartridges share a few expensive drives.
- A removable library is best devoted to the storage of infrequently used data, because the library can only satisfy a relatively small number of I/O requests per hour.

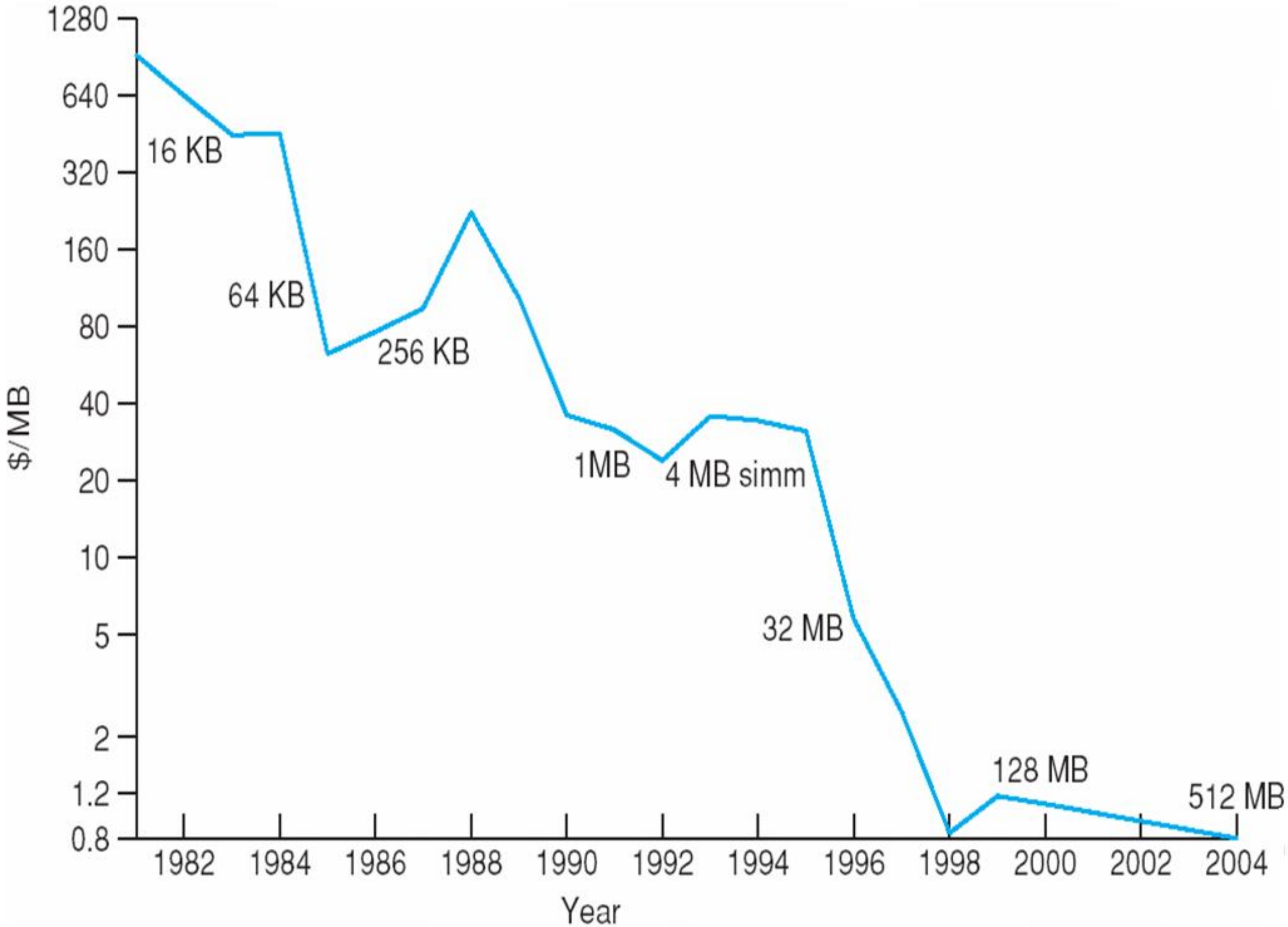
Reliability

- A fixed disk drive is likely to be more reliable than a removable disk or tape drive.
- An optical cartridge is likely to be more reliable than a magnetic disk or tape.
- A head crash in a fixed hard disk generally destroys the data, whereas the failure of a tape drive or optical disk drive often leaves the data cartridge unharmed.

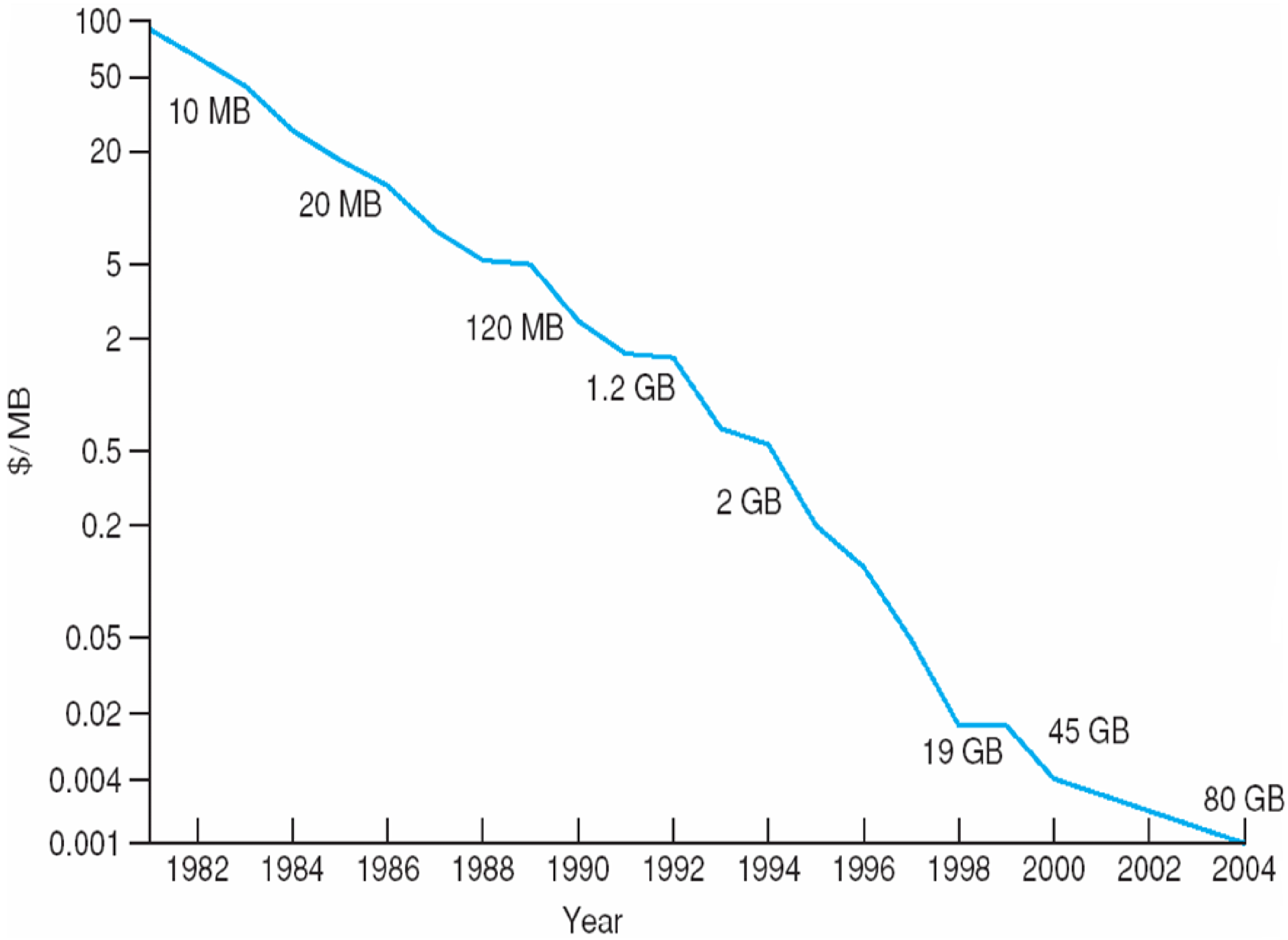
Cost

- Main memory is much more expensive than disk storage
- The cost per megabyte of hard disk storage is competitive with magnetic tape if only one tape is used per drive.
- The cheapest tape drives and the cheapest disk drives have had about the same storage capacity over the years.
- Tertiary storage gives a cost savings only when the number of cartridges is considerably larger than the number of drives.

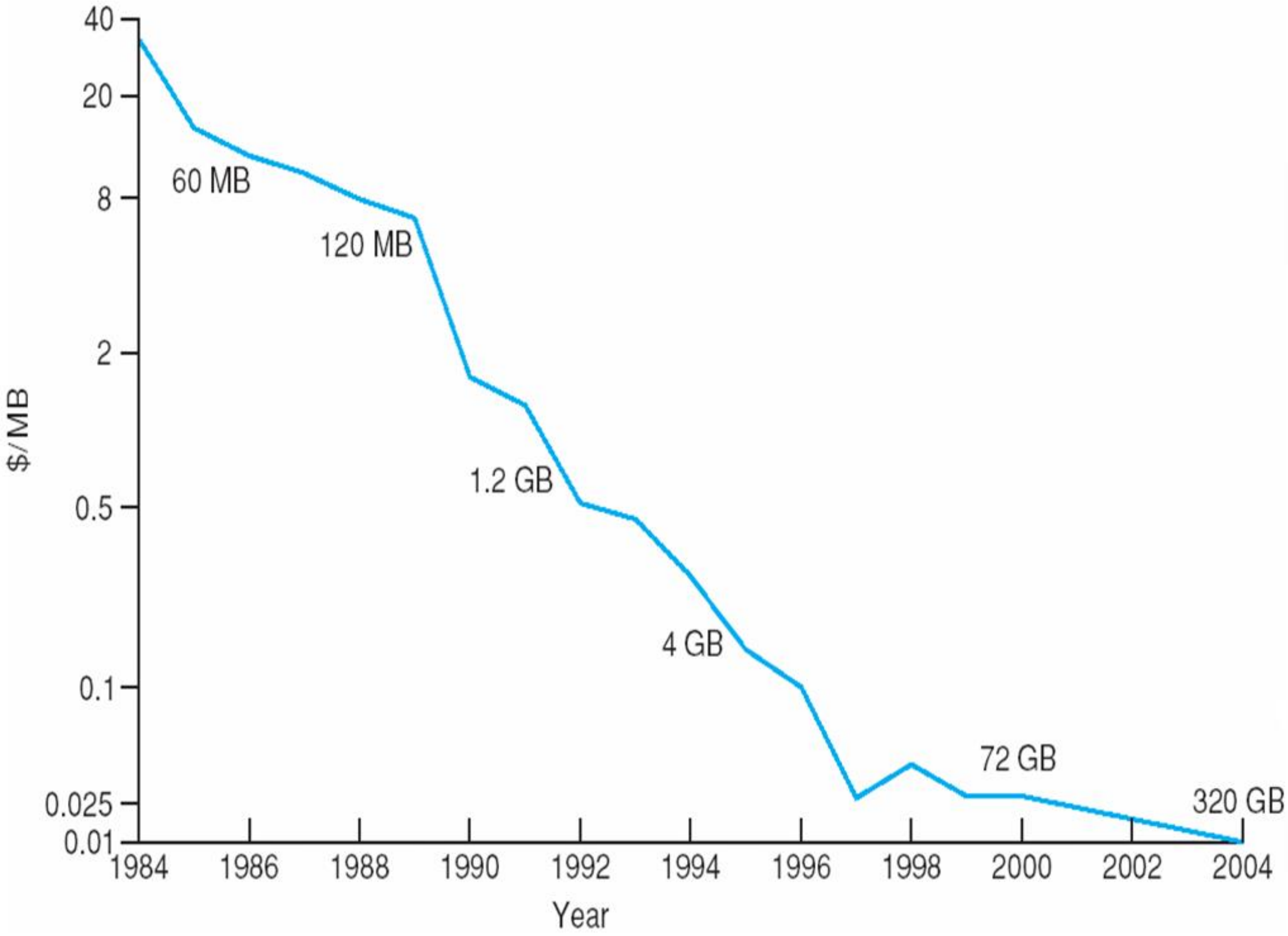
Price per Megabyte of DRAM, From 1981 to 2004



Price per Megabyte of Magnetic Hard Disk, From 1981 to 2004



Price per Megabyte of a Tape Drive, From 1984-2000



Amazon Simple Storage Service (Version 2006-03-01)

Quick Reference Card

Revised: 04/1/2010

Service Operations

GET Service

Returns a list of all buckets owned by the authenticated request sender.

```
GET / HTTP/1.1
Host: s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
```

Bucket Operations

PUT Bucket

Creates a new bucket belonging to the account of the authenticated request sender. Optionally, you can specify a EU (Ireland) or US-West (N. California) location constraint.

```
PUT / HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
Content-Length: {0 | length}
[<CreateBucketConfiguration>
  <LocationConstraint>EU</LocationConstraint>
</CreateBucketConfiguration>]
```

GET Bucket

Lists information about the objects in a bucket for a user that has read access to the bucket.

```
GET ?prefix=prefix&marker=marker&max-keys=max-keys&delimiter=delimiter HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
```

GET Bucket location

Lists the location constraint of the bucket for the bucket owner.

```
GET /?location HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
```

DELETE Bucket

Deletes the specified bucket. All objects in the bucket must be deleted before the bucket itself can be deleted.

```
DELETE / HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
```

POST Object

For more information about POST, refer to the *Amazon Simple Storage Service Developer Guide.*

Object Operations

GET Object

Gets an object for a user that has read access to the object.

```
GET /destinationObject HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
[Range:bytes=byte_range]
[x-amz-metadata-directive: metadata_directive]
[x-amz-if-match: etag]
[x-amz-if-none-match: etag]
[x-amz-if-unmodified-since: time_stamp]
[x-amz-if-modified-since: time_stamp]
```

PUT Object

Adds an object to a bucket for a user that has write access to the bucket. A success response indicates the object was successfully stored; if the object already exists, it will be overwritten.

```
PUT /destinationObject HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
Content-Length: length
Content-MD5: md5_digest
Content-Type: type
Content-Disposition: object_information
Content-Encoding: encoding
Cache-Control: caching
Expires: expiration
<request metadata>
```

COPY Object

Copies an object for a user that has write access to the bucket and read access to the object. All headers prefixed with x-amz- must be signed, including x-amz-copy-source.

```
PUT /destinationObject HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
x-amz-copy-source: /source_bucket/sourceObject
[x-amz-metadata-directive: metadata_directive]
[x-amz-copy-source-if-match: etag]
[x-amz-copy-source-if-none-match: etag]
[x-amz-copy-source-if-unmodified-since:
time_stamp]
[x-amz-copy-source-if-modified-since: time_stamp]
<request metadata>
```

HEAD Object

Retrieves information about an object for a user with read access without fetching the object.

```
HEAD /destinationObject HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
```

DELETE Object

Deletes the specified object. Once deleted, there is no method to restore or undelete an object.

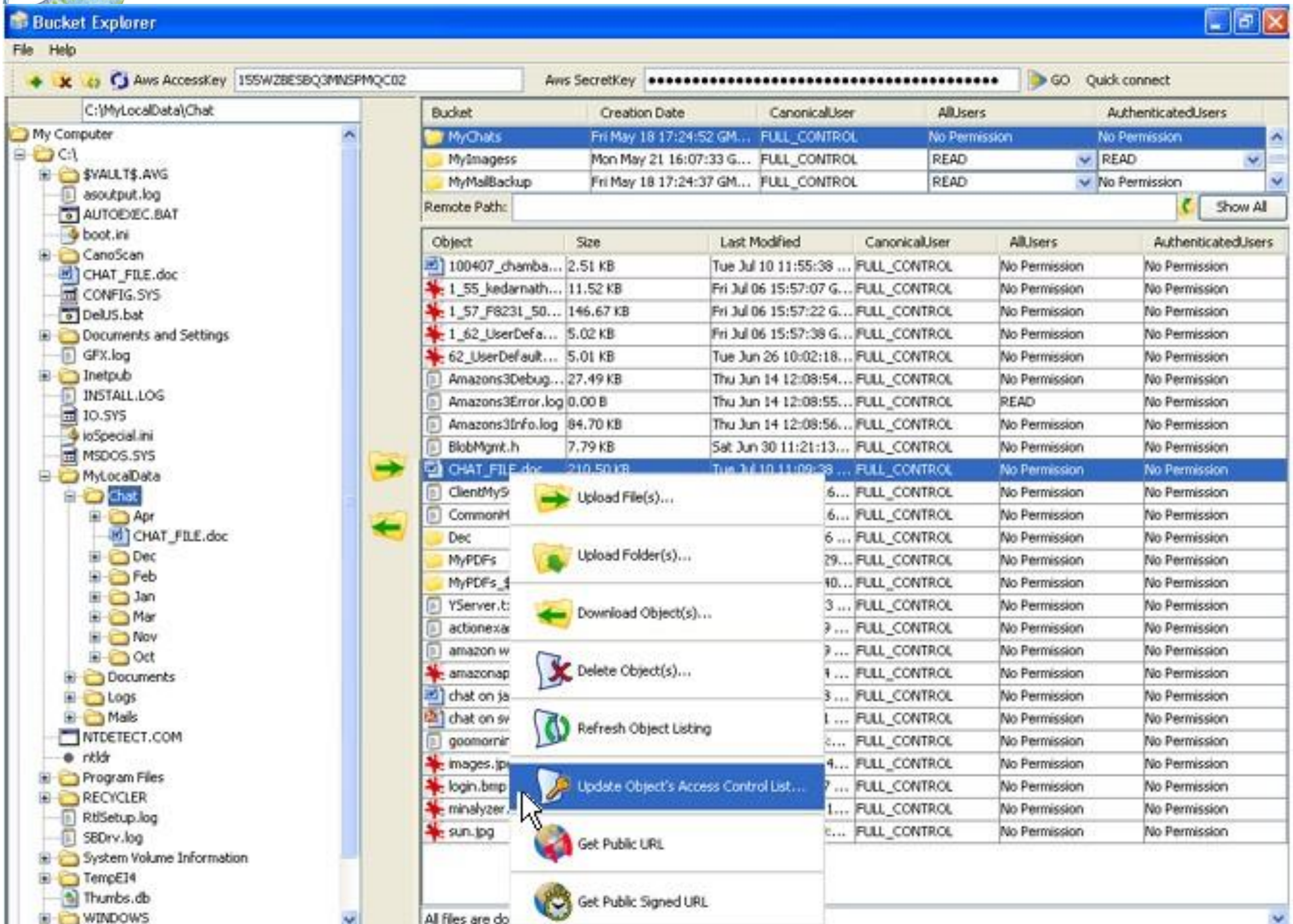
```
DELETE / HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
Date: date
Authorization: AWS AWSAccessKeyId:signature
```

POST Object

Adds an object to a bucket using forms.

```
POST / HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
User-Agent: browser_data
Accept: file_types
Accept-Language: locales
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=----------ID
Content-Length: length
<multiform_data>
```

Note: For more information about the multiform data, refer to the *Amazon Simple Storage Service Developer Guide.*


Bucket Explorer
File Help
Aws AccessKey 15SWZBESBQ3MNSPMQC02
Aws SecretKey
GO
Quick connect
C:\MyLocalData\Chat
My Computer
C:\
Bucket
Creation Date
CanonicalUser
AllUsers
AuthenticatedUsers
MyChats
MyImagess
MyMailBackup
Remote Path:
Show All
Object
Size
Last Modified
Upload File(s)...
Upload Folder(s)...
Download Object(s)...
Delete Object(s)...
Refresh Object Listing
Update Object's Access Control List...
Get Public URL
Get Public Signed URL

The whole internet 'weighs the same as a strawberry'

UPDATED: 17:28 GMT, 4 November 2011

Share Tweet +1 Share 31 View comments

A mathematician recently calculated that eBook readers 'gain weight' when you add new books to your library - due to the energy 'gained' by electrons when they store information, and the weight of that energy.

Filling a Kindle with books causes it to gain an infinitesimally small amount of mass - so small that it gains 100,000,000 times more when you recharge the battery.

Now a YouTube science channel has used the same mathematics to calculate the mass of the entire internet.

Surprisingly, the whole thing weighs just 50g - around the weight of a single (large) strawberry.

The entire internet weighs as much as a strawberry, calculates YouTube science show Vsauce. But if you're only counting the data, not the electricity required to make it work, the whole lot weighs far far less

End of Chapter 12