6. Media Servers

6.1 Media Server Architecture

Media server

- A special type of data / file server
- High-volume data transfer
- Real-time access
- Large files (objects, data units)

Pull model

- The client controls the data delivery
- Suitable for editing of content over networks

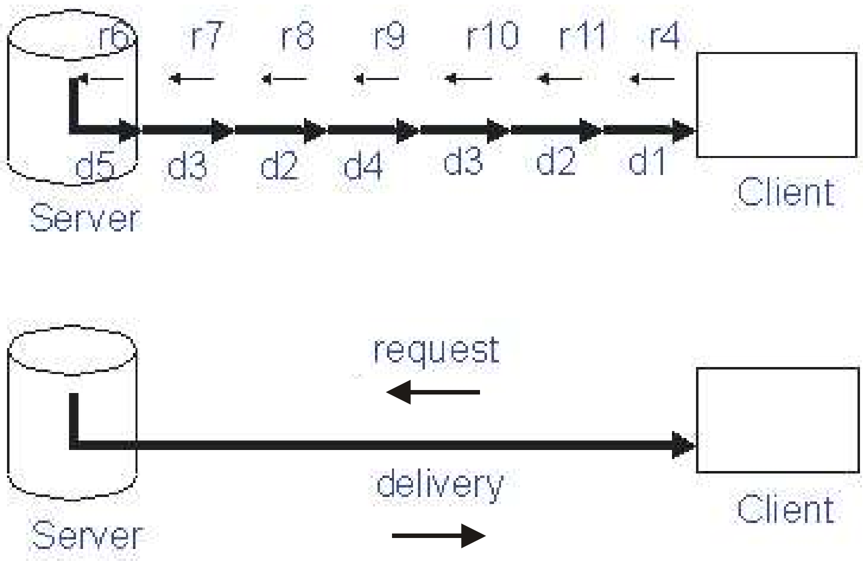
Push model

- Also known as “data pump“
- The server controls data delivery
- Suitable for broadcasting data over networks

Basic models: pull & push

- Mainly an application point of view how to interact with media data
- Mixtures possible: application sends “play list” to server
- Same server internals apply to both models (i.e., not treated separately in the rest of this chapter)

Media Server - Push and Pull Model



Media Server Architecture Components

- **Network attachment**
  - typically a network adapter
- **Content directory**
  - responsible for verifying if content is available on the media server, and
  - if the requesting client is allowed to access the data
- **Memory management**
  - caching for large amounts of data and performance improvement
- **File system**
  - handles the organization of the content on the server
  - this includes assignment of sufficient storage space during the upload phase
- **Storage management**
  - abstraction of driver
  - responsible for disk scheduling policies and layout of files
- **Disk controller**
  - handles access to data on the storage device
  - head movement speed, I/O bandwidth, the largest and smallest units that can be read at a time, and the granularity of addressing, (e.g., RAID)

Scaling of a Media Server - Cluster of Servers (1)

Motivation

- Growth of systems implies replication of multiple components

Approach

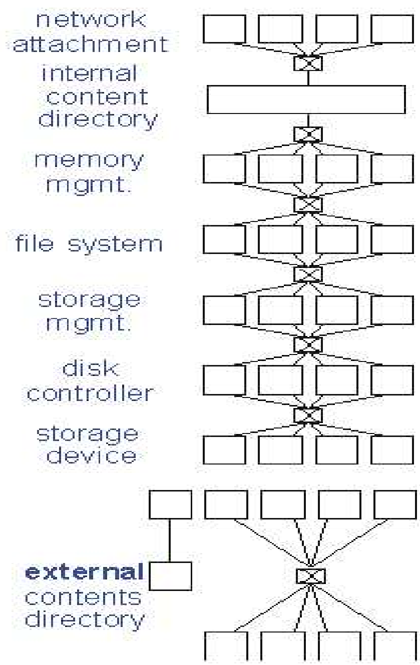
- Optimization of each component
- Distributed onto possibly heterogeneous components
- Cooperation between distributed components

Issues to be solved

Example: Content directory must always be consistent

- Internal content directory, once per media server
- External content directory

Cluster of Servers (2)



6.2 Storage Devices and Disk Layout

Tape

- Cannot provide multiple streams in parallel
- Random access is slow

Disk

- Access times: Seek time typically 8 ms on a magnetic disk vs. 150 ms on an optical disk
- CLV vs. CAV:
  - Magnetic disks usually have constant rotational speed, thus
    - constant angular velocity, CAV
    - more space on outside tracks
  - Optical disks have varying rotational speed
    - constant linear velocity, CLV
    - same storage capacity on inner and outer tracks
- Capacity vs. cost: Optical cheaper than magnetic
- Type of persistence (Rewritable, Write-once, Read-only, e.g., CD-ROM)

Disk Layout (1)

Determines

- the way in which content is addressed
- how much storage space on the media is actually addressable and usable
- the density of stored content on the media

Multiple track vs. single track (CD)

- changes on single track data are expensive

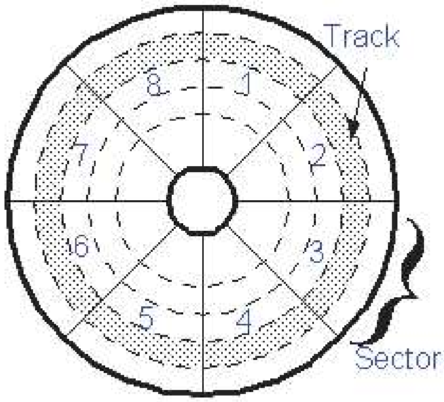
Tracks and sectors

- access restricted to the unit of a sector
- unused space of a sector is wasted

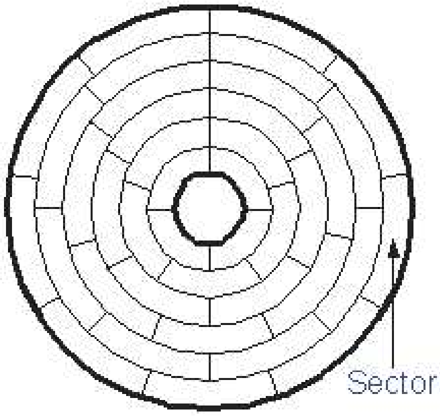
Zone Bit Recording

- motivation: a sector at an outer radius has the same (sector) data amount, but more raw capacity
- constant angular velocity
- i.e. same access time to inner/outer tracks
- different read/write speeds, depending on radius
- Can be used to place more popular media (movies) on an outer track, less popular media on an inner track. This saves disk arm movements.

Disk Layout (2)



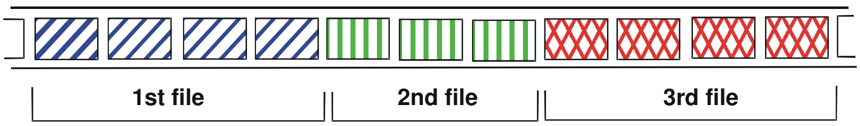
(a) CAV, traditional recording



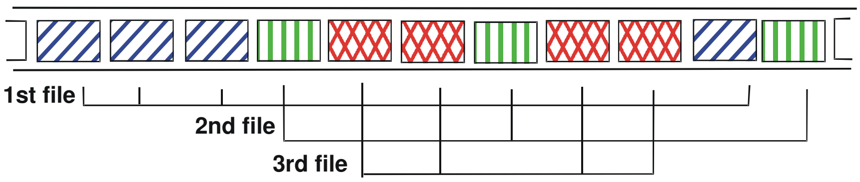
(b) CAV, zone bit recording

Placement of Files at Storage Device Level(1)

contiguous placement:



non-contiguous placement:



Placement of Files at Storage Device Level(2)

A file is a sequence of bytes with a special “end of file” symbol.

- **Contiguous (sequential) placement:** stored in the order in which it will be read
  - like on a tape
  - fewer seek operations during playback, i.e., good for “continuous” access
  - less flexibility, problematic when data needs to be changed.
- **Non-contiguous placement**, i.e. scatter blocks across disk:
  - avoids external fragmentation (“holes” between contiguous files)
  - same data can be used for several streams via references
  - long seek operations during playback

6.3 Disk Controller and RAID

RAID = Redundant Array of Inexpensive Disks

Motivation

- Sometimes better to provide a set of disks instead of one large disk, i.e. for “striping“

Goals: to enhance storage size AND

- primarily: fault tolerance (availability, stability)
  - by redundancy
  - related to (as low as possible) additional expenses
- secondarily: performance
  - by data striping
    - by distributing data transparently over multiple disks and making them appear as a single fast disk
  - fast read and write
  - for small and large amounts of data

RAID and multimedia

- RAID can help to improve multimedia data delivery from servers

Redundant Array of Inexpensive Disks

Granularity of data interleaving

- fine grained
  - small units to be interleaved
  - any I/O request (regardless of data unit size) involves all disks
- coarse grained
  - larger units of data to be interleaved
  - a small file (total data request) may involve only some disks

Method and pattern of placing redundant data

- Computing redundancy data: most often parity, sometimes Hamming or Reed-Solomon codes
- Distribution/placement
  - either concentrate redundancy on some disks
  - or distribute it uniformly

Reference

E.g., Chen et al: RAID: High-Performance, Reliable Secondary Storage, ACM Computing Surveys, Vol. 26, No. 2, June 1994

Non-Redundant (RAID Level 0) (1)

Goal and Usage

- to enhance pure I/O performance
- Mainly for use in supercomputers

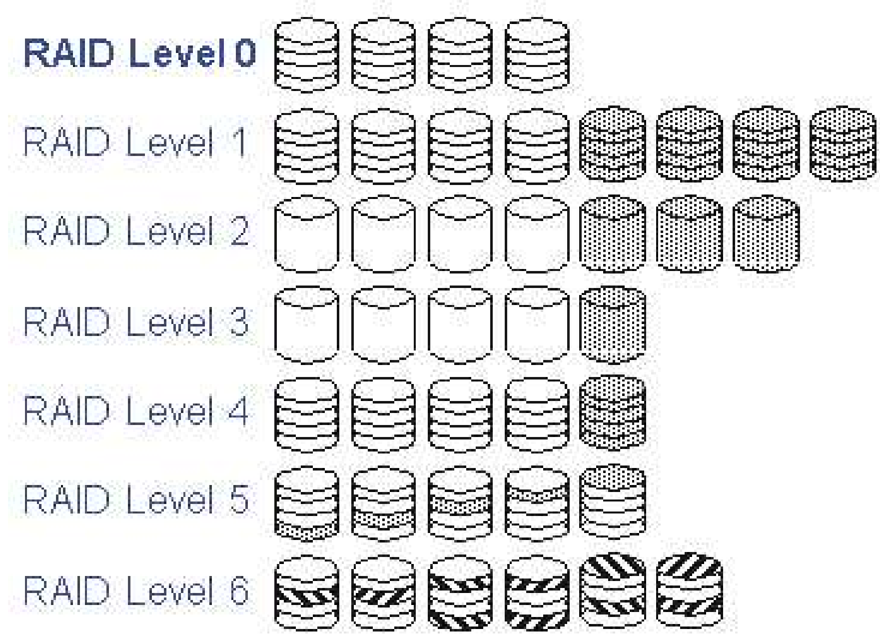
Approach

- data striping among a set of e.g. 4 disks
- **A block of data is split**, different parts of it are stored on different devices
- 4 disks of 1 GB provide in total a capacity of 4 GB
- Implementation
  - i.e., SCSI allows for up to 8 daisy-chained controllers and up to 56 logical units

Performance

- read
  - very good but mirrored disks may be better (if appropriate schedules are used)
- write
  - best of all RAID performances (no need to update redundant data)

Non-Redundant (RAID Level 0) (2)



Mirrored (RAID Level 1) (1)

Goal and Usage

- better fault tolerance
- frequently used for databases (when availability is more important than storage efficiency)

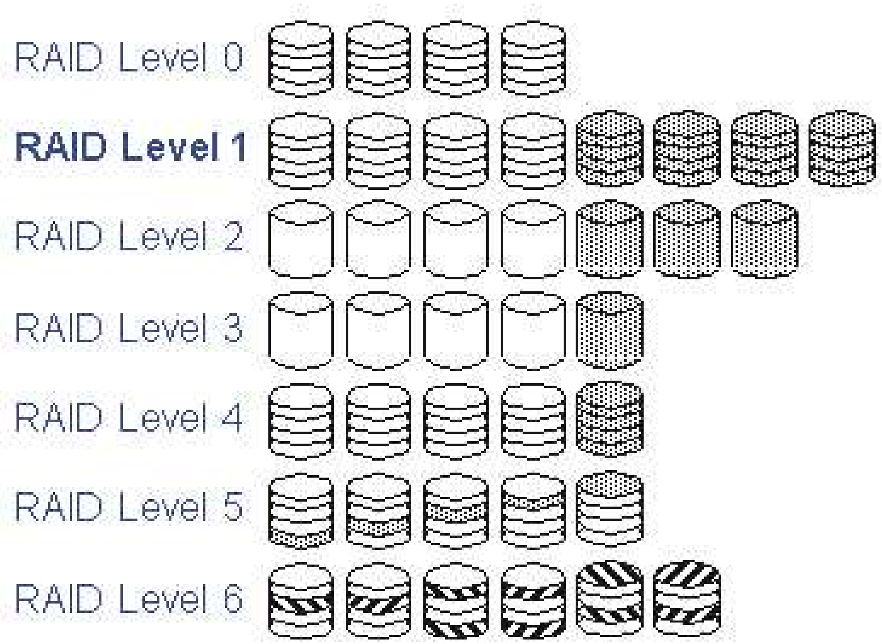
Approach

- **Mirrored disks** (or shadowing): duplicate data is written to a second disk
- Every sector on the primary disk is also stored on the secondary disk

Performance

- read
  - parallel reads can increase the I/O performance, or the disk with shorter queues / rotational delay / seek time can be selected
  - if different controllers are used
- write
  - slowed down (write must be done on two devices simultaneously)

Mirrored (RAID Level 1) (2)



Memory-Style ECC (RAID Level 2) (1)

Goal

- to enhance fault tolerance
- to reduce RAID level 1 hardware costs

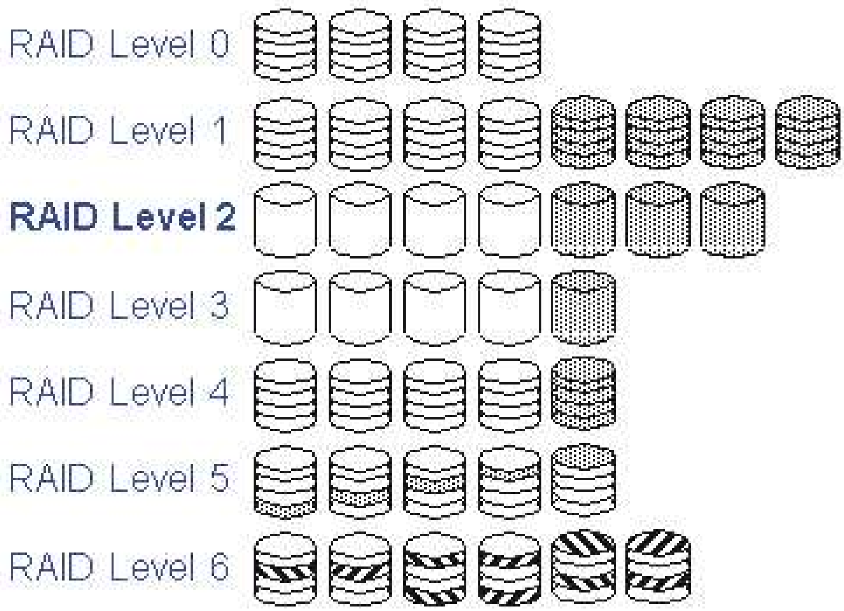
Approach

- **Bit striping among various disks with additional error correction codes (Hamming codes) on separate disks**
- error detection:
  - single parity disk
- but here error correction used is proportional to log (number of disks)
  Example 1: 10 data disks with 4 parity disks
  Example 2: 23 data disks and 5 parity disks

Performance

- minimum amount of data that must be transferred is related to the number of disks (one sector on each disk)
- large amount leads to better performance
- slower disk recovery

Memory-Style ECC (RAID Level 2) (2)



Bit-Interleaved Parity (RAID Level 3) (1)

Goal and use

- to enhance fault tolerance
- to reduce RAID level 2 hardware costs
- application when
  - high bandwidth is demanded
  - but not a high I/O rate

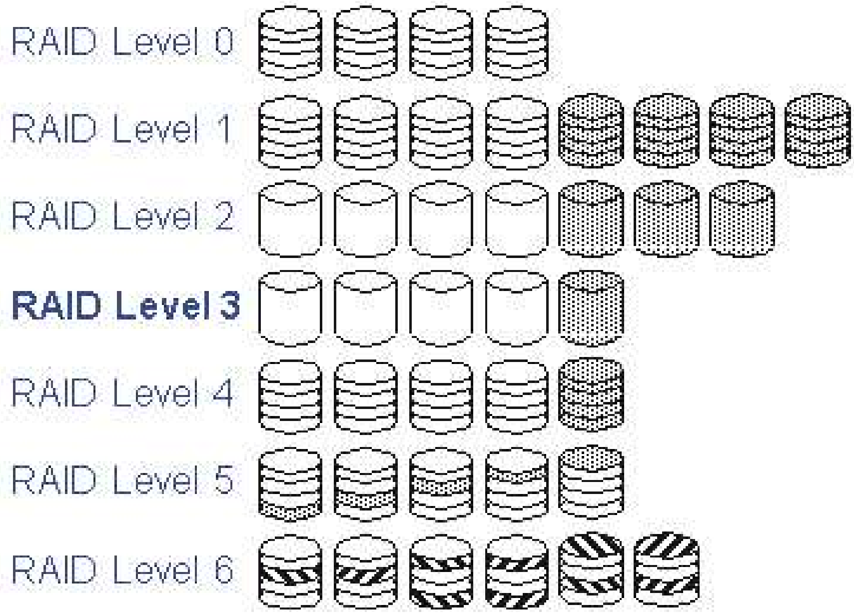
Approach

- **Bit-interleaved parity. Striping (bit-wise interleaving) across disks plus one bit parity disk**
- a single parity disk for any group/array of RAID disks; contains one parity bit for the group of data disks
- makes use of build-in CRC checks of all disks

Performance (similar to RAID level 2)

- slower disk recovery
- no interleaved I/O
- note: disks should be synchronized to reduces seek and rotational delays

Bit-Interleaved Parity (RAID Level 3) (2)



Block-Interleaved Parity (RAID Level 4) (1)

Goal

- to provide fault tolerance
- to enhance RAID level 3 performance in case of a fault

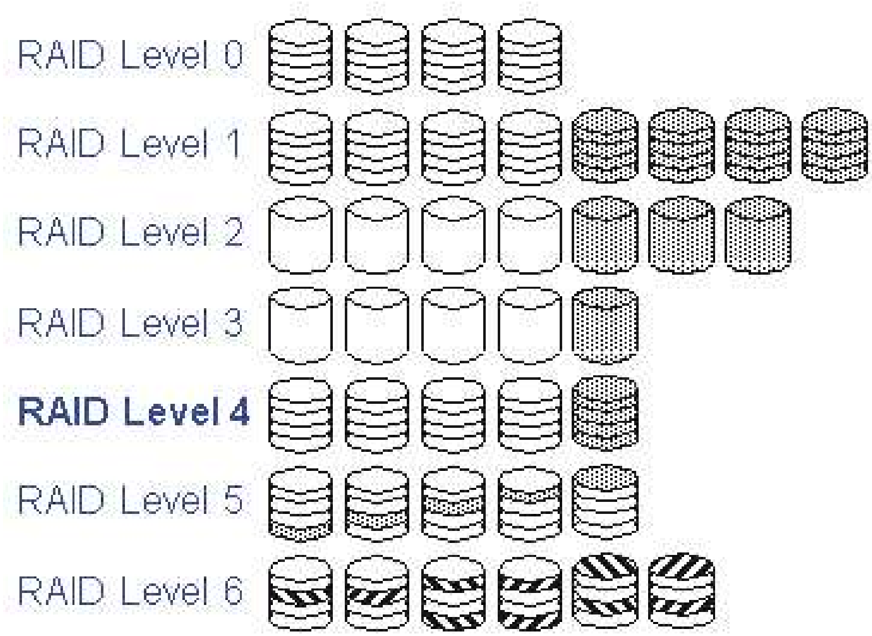
Approach

- **Block-interleaved parity. Sector striping across disks. One extra block parity disk**
- parity sectors stored on a single extra disk

Performance

- faster disk recovery possible
- small writes
  - only two disks affected (not the entire set)
  - not in parallel (only one write per disk group as parity disk is affected)
- small reads are improved
  - from one disk only
  - may occur in parallel

Block-Interleaved Parity (RAID Level 4) (2)



Block-Interleaved Distributed Parity (RAID Level 5) (1)

Goal

- to provide fault tolerance
- to remove the write bottleneck of RAID level 4

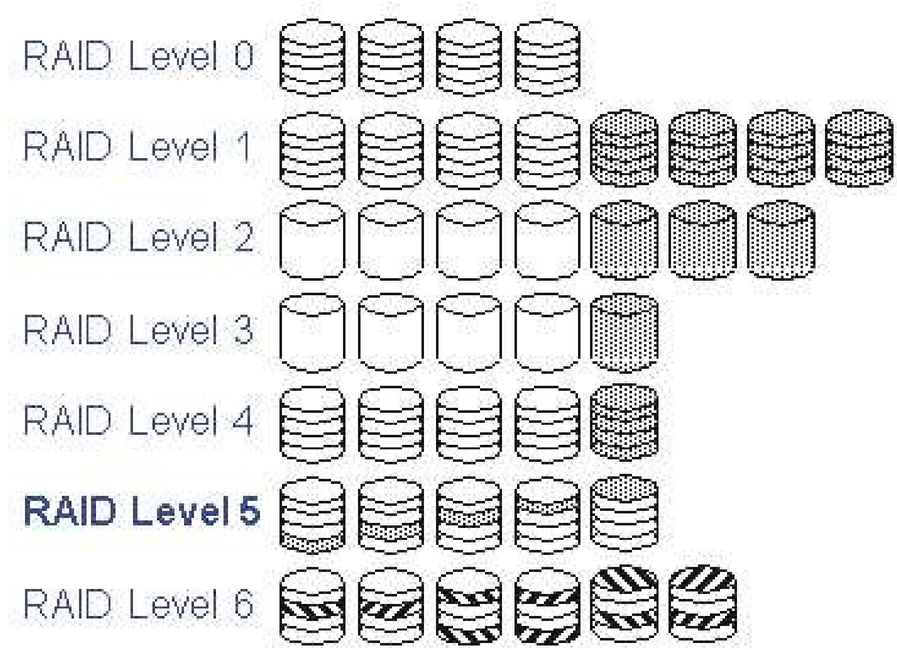
Approach

- **Sector striping across disks, parity data distributed over several disks**

Performance

- removes performance bottleneck of a single parity disk
- read and write: allow parallel operations
- small read or write
  - very good: similar to RAID level 1
- large amount of data
  - very good: similar to RAID 3 and 4

Block-Interleaved Distributed Parity (RAID Level 5) (2)



P+Q Redundancy (RAID Level 6) (1)

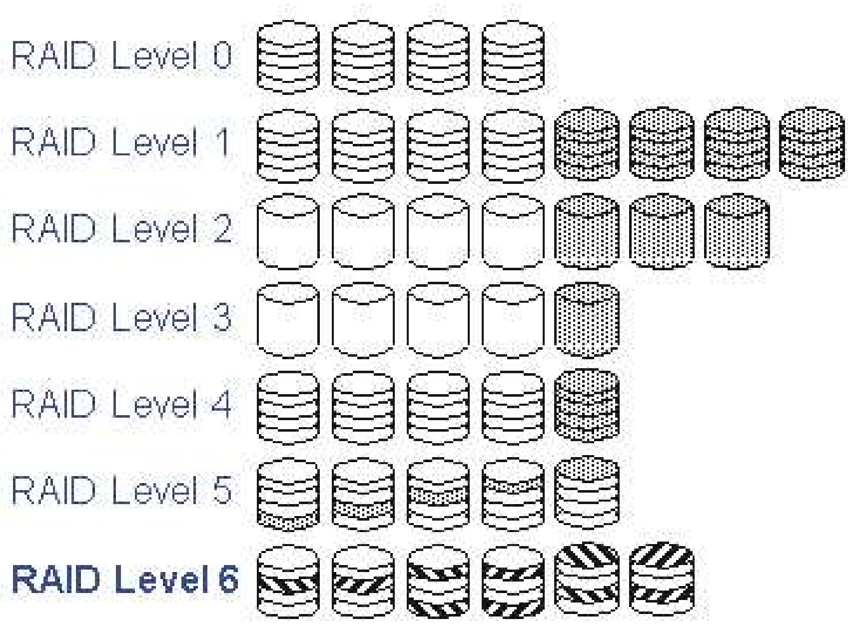
Goal

- Motivation:
  - Very large arrays may contain more than one disk with failures
- ECC is required in order to maintain availability

Approach

- ECC
  - "P+Q redundancy" based on Reed-Solomon codes
  - protects against failure of two disks at the same time
- two additional disks
- otherwise similar to RAID level 5

P+Q Redundancy (RAID Level 6) (2)



6.4 Storage Management and Disk Scheduling

Disk Management - File Placement on Disk

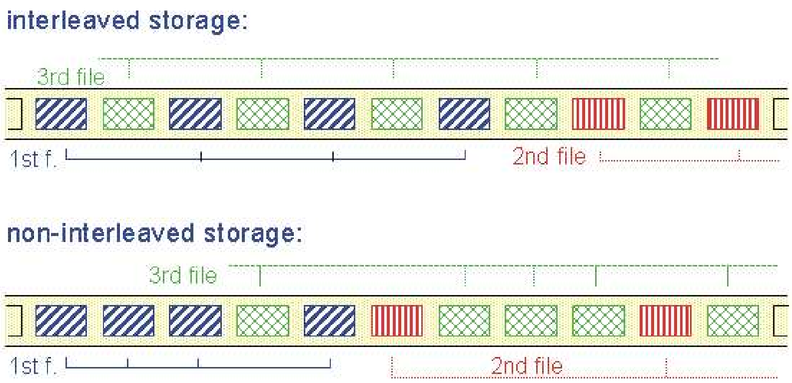
Goal: to reduce read and write times by

- fewer seek operations
- lower rotational delay or latency
- high actual data transfer rate (can not be improved by placement)

Method: store data in a specific pattern

- Regular distance
- Combine related streams
- Larger block size
  - fewer seek operations
  - smaller number of requests
  - but higher loss due to internal fragmentation (last block used only 50% on the average)

Interleaved Placement



Interleaved files

- Interleaving several streams (e.g., channels of audio)
- All nth samples of each stream are in close physical proximity on disk
- Problem: changing (inserting / deleting) parts of a stream is difficult

Interleaved vs. non-interleave and contiguous vs. non-contiguous/scattered

- Contiguous interleaved placement
- Scattered interleaved placement

Traditional Disk Scheduling

Definition:

Disk scheduling determines the order by which requests for disk access are serviced.

Disk service model

Requests are buffered and can be re-ordered before they are served by the disk.

General goals of scheduling algorithms

- Short response time
- High throughput
- Fairness (e.g., requests at disk edges should not starve)

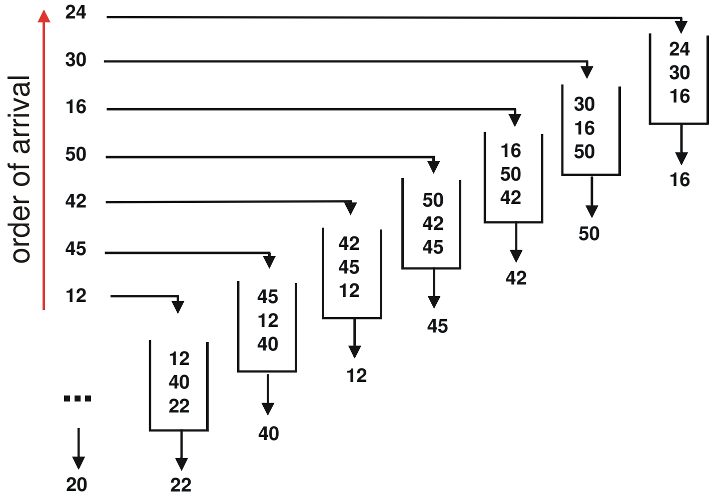
Multimedia Goals (in general)

- continuous throughput (must not be fair)
- short maximal (not average) response times
- high throughput

Typical trade-off

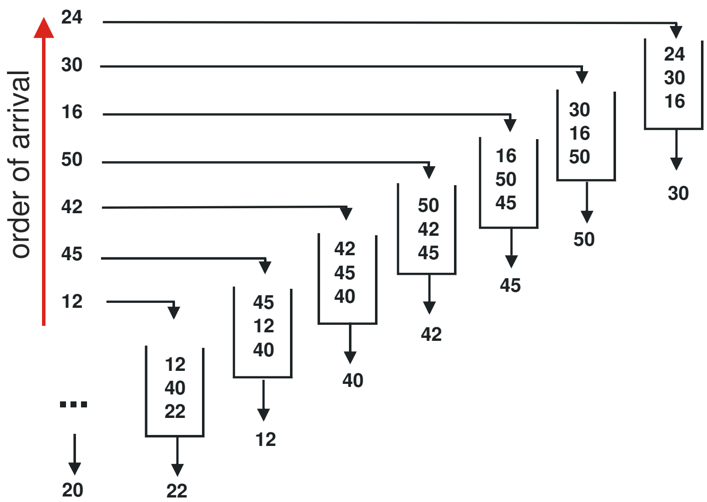
- Seek & rotational delay vs. maximum response time

First Come First Serve (FCFS) Disk Scheduling



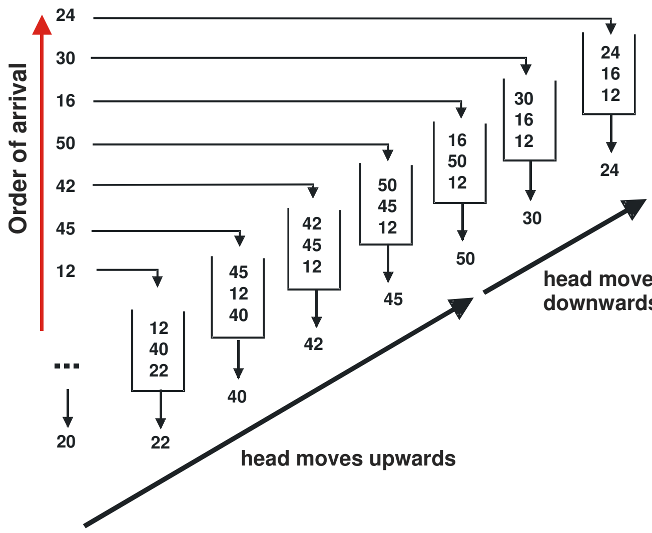
Properties
- Long seek times (since non-optimal head movement occurs)
- Short (individual) response times

Shortest Seek Time First (SSTF) Disk Scheduling



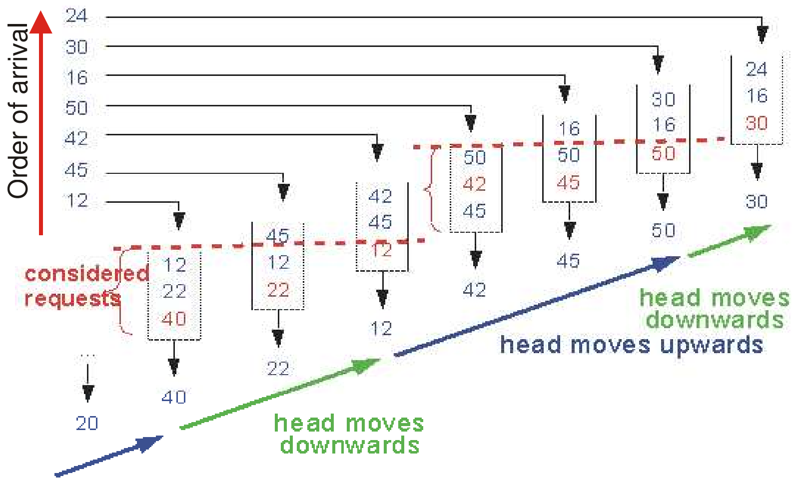
Properties
- Short seek times
- Longer maximum (individual) response times
- May lead to starvation

SCAN Disk Scheduling



- Move disk head always between disk edges (up until the end, then down until the end)
- Read next requested block in disk movement direction
- A compromise between optimization of seek times and response times
- Data in the middle of the disk has better access properties

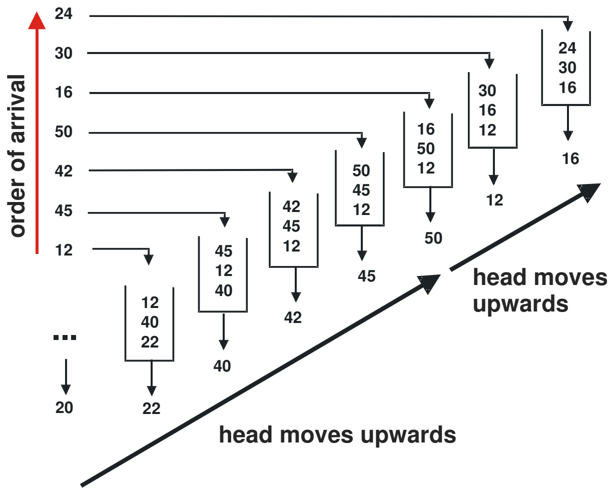
N-Step-SCAN Disk Scheduling



Properties

- reduces unfairness for outer and inner tracks
- longer seek time
- shorter response time

C-Scan Disk Scheduling



Properties

- Move disk head always between disk edges (unidirectional; up to the end, quickly down, then up to the end again)
- Improves fairness (compared to SCAN)

Disk Scheduling for Continuous Media

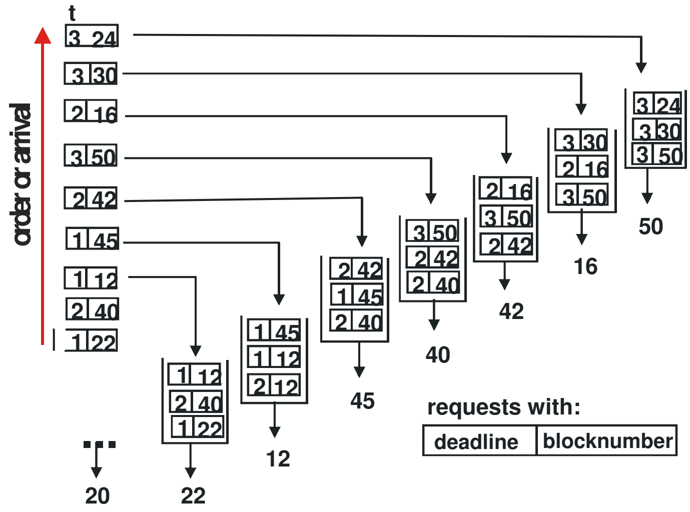
Suitability of traditional disk scheduling methods

- Effective utilization of the disk arm? short seek time
- No guaranty for / not optimized for deadlines! -> not suitable for continuous streams

Specific scheduling methods for continuous streams

- Serve continuous media, i.e., periodic requests with deadlines, plus aperiodic requests from other media
- Never miss a deadline of a continuous medium while serving aperiodic requests
- Aperiodic requests should not starve
- Provide high multiplicity (multiple streams) with real-time access
- Balance the trade-off between buffer space and efficiency

Earliest Deadline First (EDF) Disk Scheduling



Real-time scheduling algorithm

- First read the block with nearest deadline

May result in excessive seek time and poor throughput

Scan-EDF Disk Scheduling (1)

Method

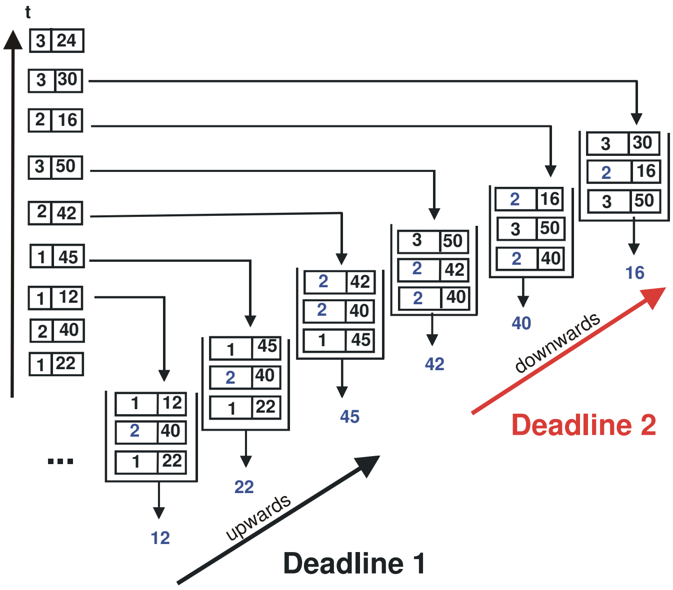
- Group requests by similar deadlines
- Requests with earlier deadlines are served first
- Among all requests with the same deadline, requests are served by track location

Combines advantages of

- SCAN (seek optimization) with
- EDF (real-time aspects)

Increases efficiency by modifying deadlines.

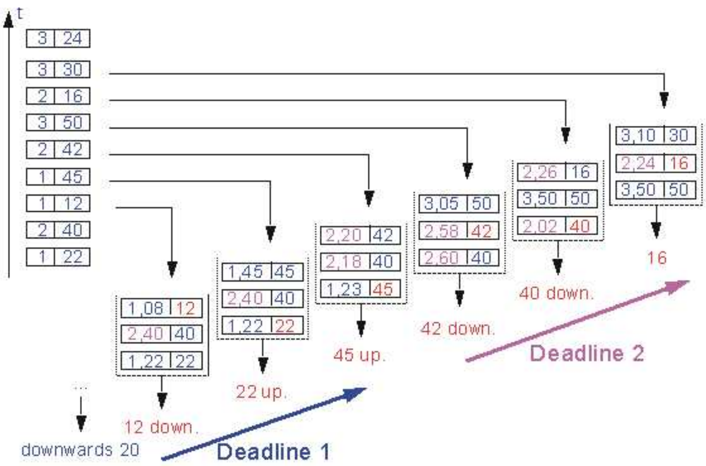
Scan-EDF Disk Scheduling (2)



Properties

- apply EDF between groups
- for all requests within a group apply SCAN

Scan-EDF Disk Scheduling: Example

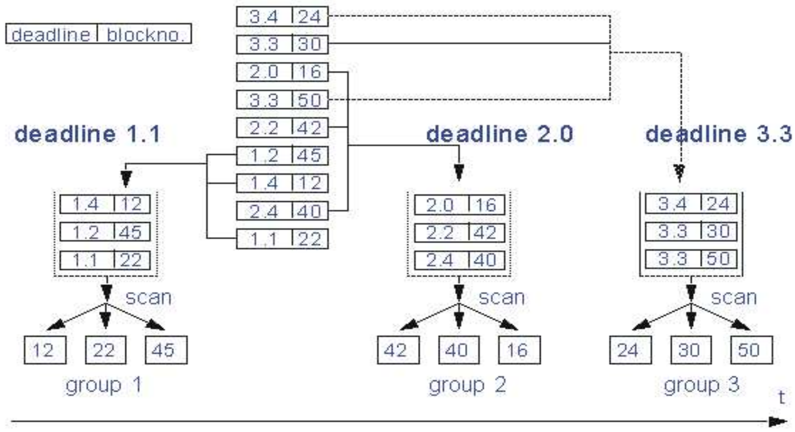


Example

At "downwards 20" the next deadlines are computed, assume N_{Max}=100

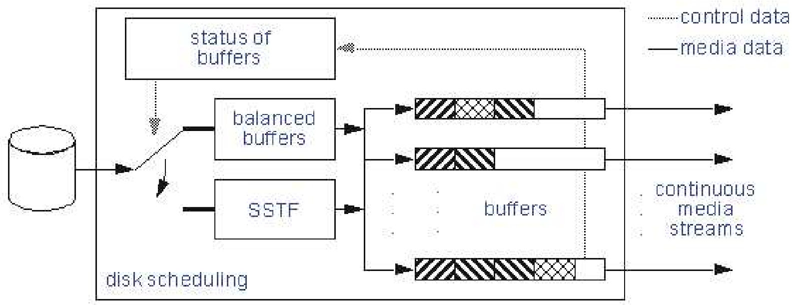
- 1 12: downwards and 12 on the way: position20 - position12 = 08, i.e. 1,08
- 2 40: downwards and 40 not on the way: =40; i.e. 2,40

Group Sweeping Scheduling



- **Form groups**
  - with deadlines lying close together
  - or in a round robin manner
- **Apply SCAN to each group**

Mixed Disk Scheduling Strategies



Goal

- maximize transfer efficiency by minimizing seek time and latency
- serve process requirements with a limited buffer space

Combines

- shortest seek time first (SSTF)
- buffer underflow and overflow prevention
- by keeping buffers filled at a similar level