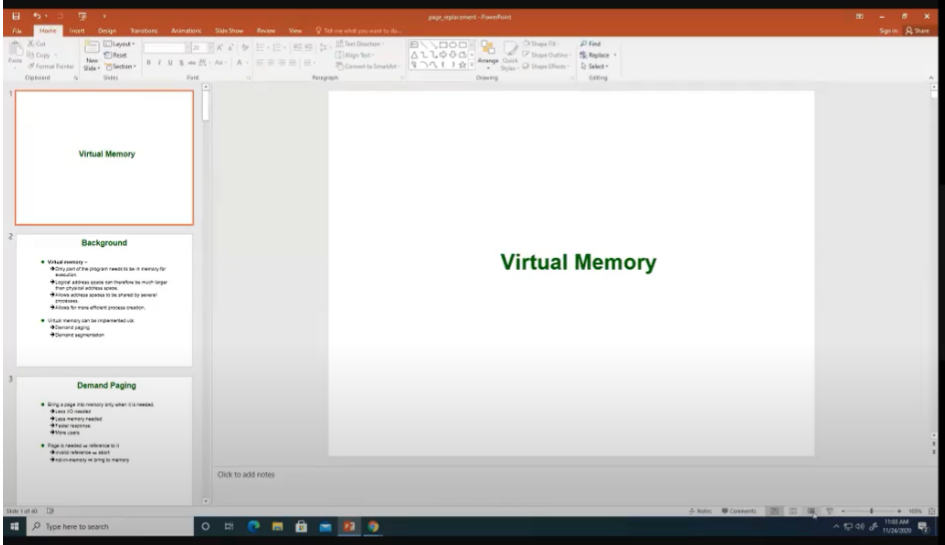
Operating systems (2020-11-23 at 21:28 GMT-8)



J P Misra

2:38 / 57:25

Operating systems (2020-11-23 at 21:28 GMT-8)

Background

- **Virtual memory** –
  - ➔Only part of the program needs to be in memory for execution.
  - ➔Logical address space can therefore be much larger than physical address space.
  - ➔Allows address spaces to be shared by several processes.
  - ➔Allows for more efficient process creation.
- Virtual memory can be implemented via:
  - ➔Demand paging
  - ➔Demand segmentation

J P Misra

3:15 / 57:25

Operating systems (2020-11-23 at 21:28 GMT-8)

Demand Paging

- Bring a page into memory only when it is needed.
  - ➔Less I/O needed
  - ➔Less memory needed
  - ➔Faster response
  - ➔More users
- Page is needed ⇒ reference to it
  - ➔invalid reference ⇒ abort
  - ➔not-in-memory ⇒ bring to memory

J

J P Misra

5:23 / 57:25

Operating systems (2020-11-23 at 21:28 GMT-8)

Valid-Invalid Bit

- With each page table entry a valid–invalid bit is associated
  (1 ⇒ in-memory, 0 ⇒ not-in-memory)
- Initially valid–invalid but is set to 0 on all entries.
- Example of a page table snapshot.

Frame #	valid-invalid bit
	1
	1
	1
	1
	0
⋮	
	0
	0

page table

- During address translation, if valid–invalid bit in page table entry is 0 ⇒ page fault.

J

J P Misra

10:12 / 57:25

Operating systems (2020-11-23 at 21:28 GMT-8)

Page Fault

- If there is ever a reference to a page, first reference will trap to OS ⇒ page fault
- OS looks at another table to decide:
  - ➔Invalid reference ⇒ abort.
  - ➔Just not in memory.
- Get empty frame.
- Swap page into frame.
- Reset tables, validation bit = 1.
- Restart instruction:

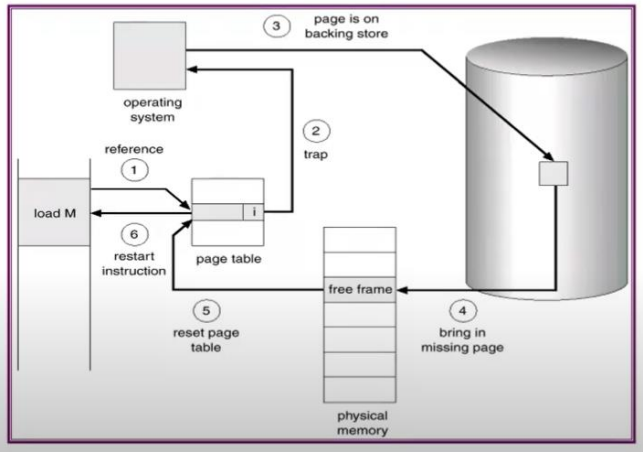
J

J P Misra

11:59 / 57:25

Operating systems (2020-11-23 at 21:28 GMT-8)

Steps in Handling a Page Fault

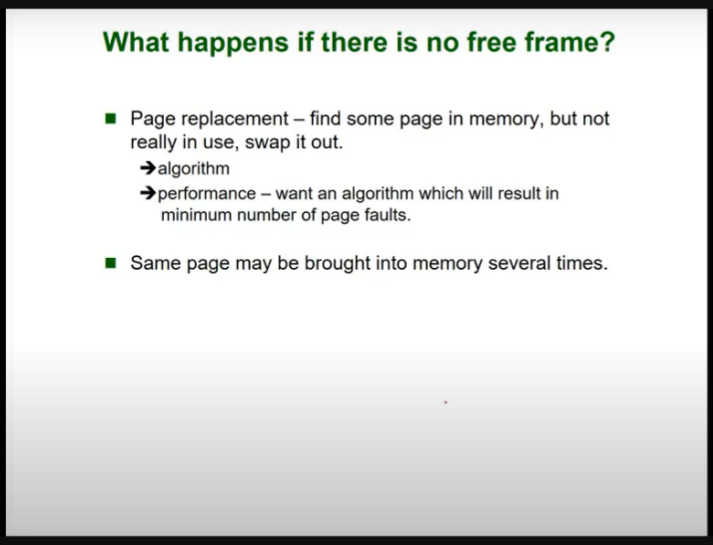


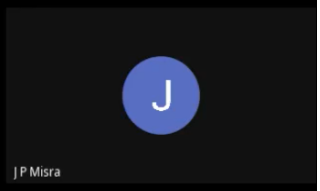
J

J P Misra

15:57 / 57:25

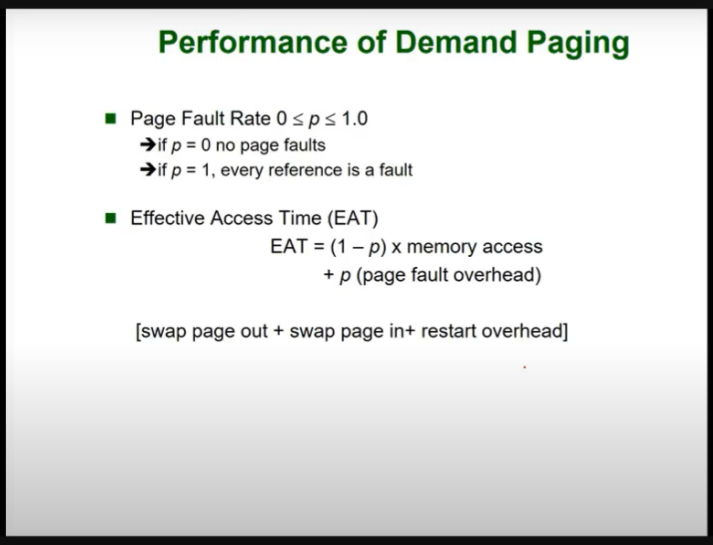
Operating systems (2020-11-23 at 21:28 GMT-8)

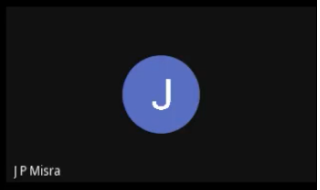




20:49 / 57:25

Operating systems (2020-11-23 at 21:28 GMT-8)





24:35 / 57:25

Operating systems (2020-11-23 at 21:28 GMT-8)

Demand Paging Example

- Memory access time = 1 microsecond
- 50% of the time the page that is being replaced has been modified and therefore needs to be swapped out.
- Swap Page Time = 10 msec = 10,000 microsec

$$EAT = (1 - p) \times 1 + p\ (10000)$$
$$1 + 10000p$$

J

J P Misra

27:13 / 57:25

Page Replacement

- Page-fault service routine includes page replacement.
- Use *modify* (*dirty*) *bit* to reduce overhead of page transfers – only modified pages are written to disk.
- Page replacement completes separation between logical memory and physical memory
  - large virtual memory can be provided on a smaller physical memory.

J

J P Misra

Replacement Policy

- Which page to replaced?
- Page removed should be the page least likely to be referenced in the near future
- Most policies predict the future behavior on the basis of past behavior

J

J P Misra

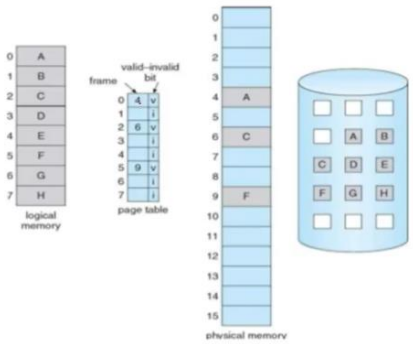
Replacement Policy

- Frame Locking
  - ➔If frame is locked, it may not be replaced
  - ➔Kernel of the operating system
  - ➔Key control structures
  - ➔I/O buffers
  - ➔Associate a lock bit with each frame

J

J P Misra

Page table when some pages are not in memory



J P Misra

What is Page fault ?

- When the data (page) requested by program is not present in main memory , it is known as page fault
- Why page replacement required ?
  - System may not have enough RAM to store all required data

J P Misra

Page Replacement Algorithms

- Want lowest page-fault rate.
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.
- In all our examples, the reference string is

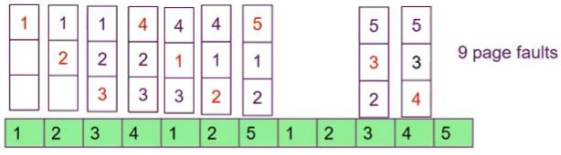
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

J

J P Misra

First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)



9 page faults

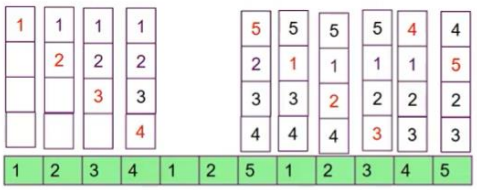
J

J P Misra

First-In-First-Out (FIFO) Algorithm

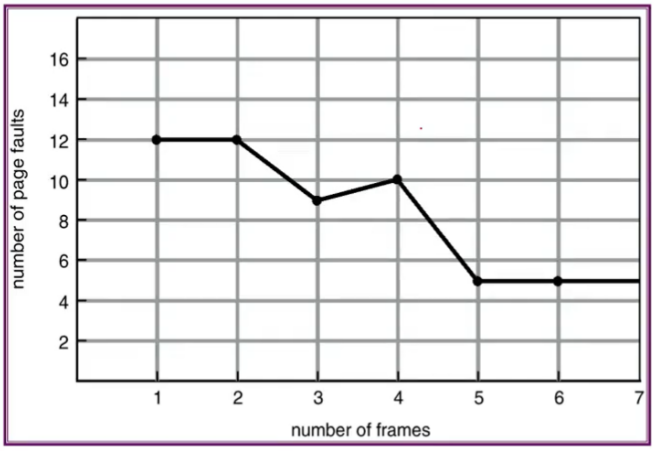
- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 4 frames (4 pages can be in memory at a time)
- In general more frames ⇒ less page faults
- FIFO Replacement – Belady's Anomaly

10 page faults



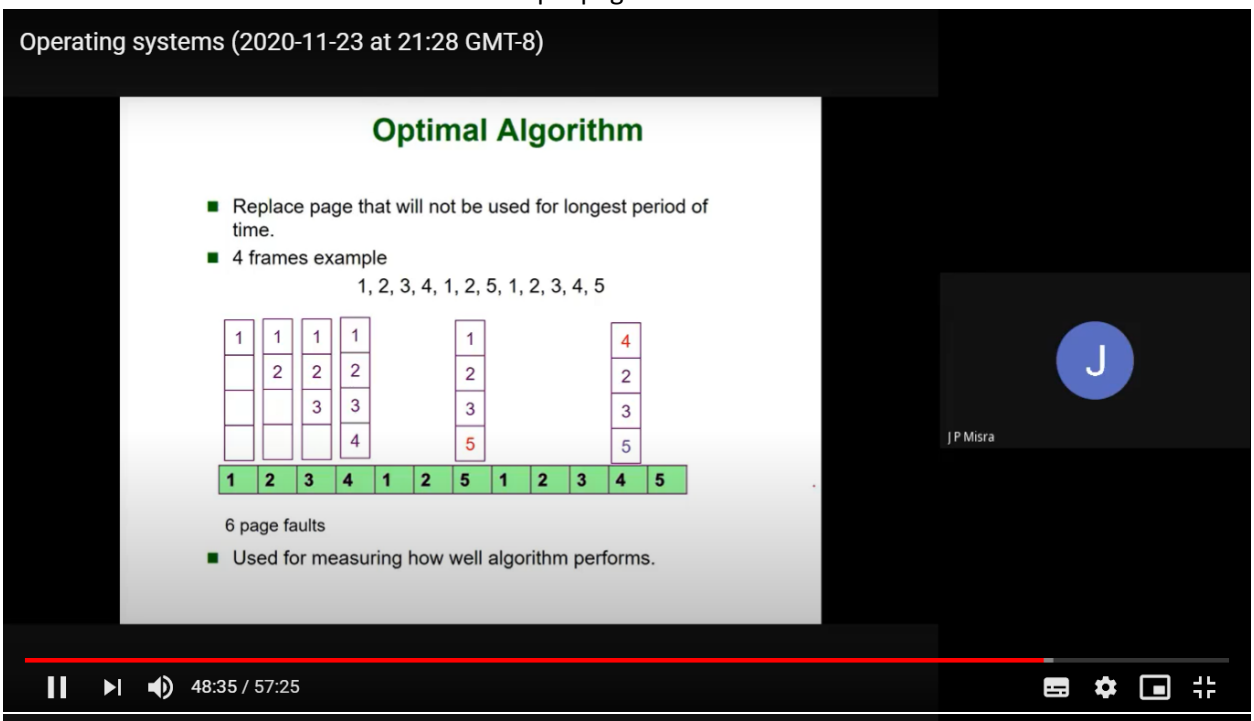
J P Misra

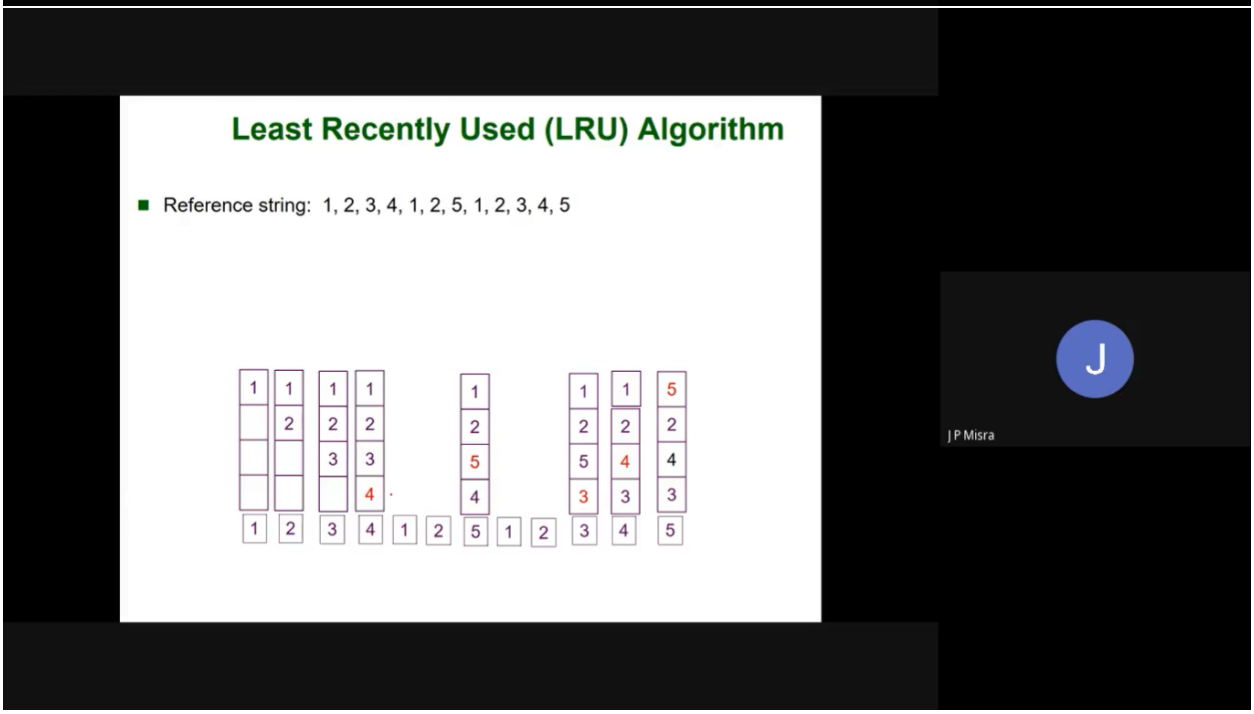
FIFO Illustrating Belady's Anomaly



J P Misra

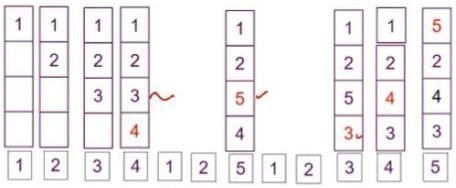
Minimize # of PFs when # frames >= # of unique pages





Least Recently Used (LRU) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



J P Misra

LRU Implementation

- Counter implementation
  - ➔Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter.
  - ➔When a page needs to be replaced , look at the counters to determine which page to replace .

J P Misra

LRU Algorithm (Cont.)

- Stack implementation
  - keep a stack of page numbers in a double link form:

➔whenever Page is referenced:

➔move it to the top

➔No search for replacement

J

J P Misra

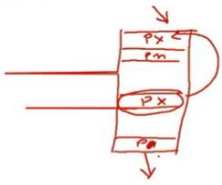
LRU Algorithm (Cont.)

- Stack implementation
  - keep a stack of page numbers in a double link form:

➔whenever Page is referenced:

➔move it to the top

➔No search for replacement



J

J P Misra

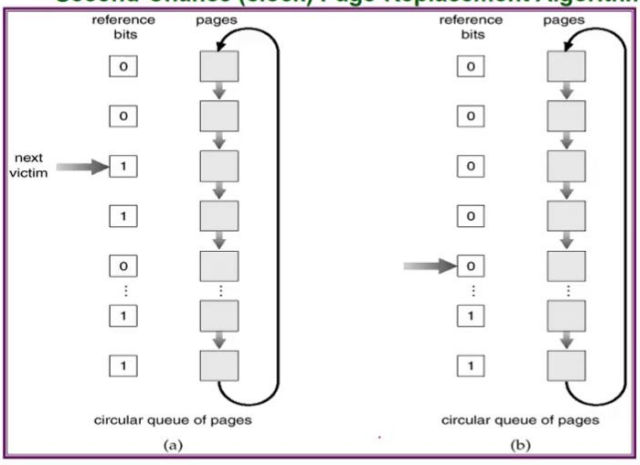
LRU Approximation Algorithms

- Reference bit
  - ➔With each page associate a bit, initially = 0
  - ➔When page is referenced bit set to 1.
  - ➔Replace the one which is 0 (if one exists). We do not know the order, however.
- Second chance
  - ➔Need reference bit.
  - ➔If page to be replaced (in clock wise order) has reference bit = 1. then:
    - ➔set reference bit 0.
    - ➔leave page in memory.
    - ➔replace next page (in clock wise order), subject to same rules.

J

J P Misra

Second-Chance (clock) Page-Replacement Algorithm



J

J P Misra

Enhanced Clock Policy

- In addition to reference bit use modify bit also
  - (0 ,0) not referenced not modified
  - (0, 1) Not recently used but modified
  - (1,0) recently used but not modified
  - (1,1) recently used and modified

J

J P Misra

Comparison

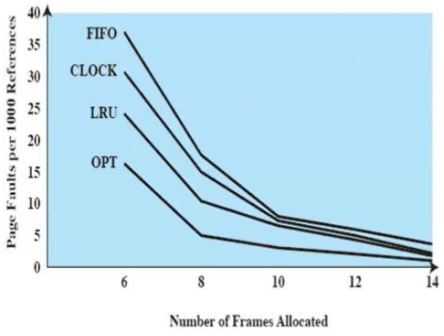


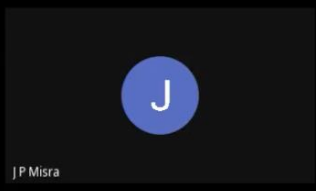
Figure 8.17 Comparison of Fixed-Allocation, Local Page Replacement Algorithms

J

J P Misra

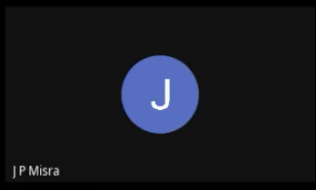
Counting Algorithms

- Keep a counter of the number of references that have been made to each page.
- LFU Algorithm: replaces page with smallest count.
- MFU Algorithm: based on the argument that the page with the smallest count was probably just brought in and has yet to be used.



Allocation of Frames

- Each process needs **minimum** number of pages.
- Example:
- MOV source, destination
  - ➔instruction is 4 bytes, might span 2 pages.
  - ➔2 pages to handle **from**.
  - ➔2 pages to handle **to**.
- Two major allocation schemes.
  - ➔fixed allocation
  - ➔priority allocation



Fixed Allocation

- Equal allocation – e.g., if 100 frames and 5 processes, give each 20 pages.
- Proportional allocation – Allocate according to the size of process.

s_i = size of process p_i

$S = \sum s_i$

m = total number of frames

a_i = allocation for $p_i = \frac{s_i}{S} \times m$

$m = 64$

$s_1 = 10$

$s_2 = 127$

$a_1 = \frac{10}{137} \times 64 \approx 5$

$a_2 = \frac{127}{137} \times 64 \approx 59$

J P Misra

OS lect (2020-11-25 at 03:28 GMT-8)

Priority Allocation

- Use a proportional allocation scheme using priorities rather than size.
- If process P_i generates a page fault,
  - ➔select for replacement one of its frames.
  - ➔select for replacement a frame from a process with lower priority number.

J P Misra

44:34 / 55:42

Global vs. Local Allocation

- **Global** replacement – process selects a replacement frame from the set of all frames; one process can take a frame from another.
  - ➔ Process cannot control its own Page fault rate
- **Local** replacement – each process selects from only its own set of allocated frames.
  - ➔Number of frames allocated to a process do not change
  - ➔Does not make use of less used pages belonging to other processes

J

J P Misra

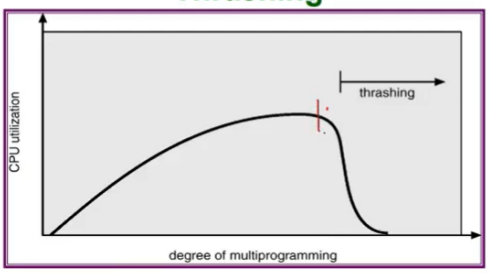
Thrashing

- If a process does not have "enough" pages, the page-fault rate is very high. This leads to:
  - ➔low CPU utilization.
  - ➔operating system thinks that it needs to increase the degree of multiprogramming.
  - ➔another process added to the system.
- **Thrashing** ≡ a process is busy swapping pages in and out.
  - ➔ More pronounced for Global page replacement policy

J

J P Misra

Thrashing



- Why does paging work?
  Locality model
  - ➔Process migrates from one locality to another.
  - ➔Localities may overlap.
- Why does thrashing occur?
  Σ size of locality > total memory size

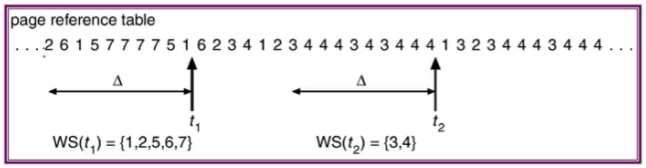
J P Misra

Working-Set Model

- Δ ≡ working-set window ≡ a fixed number of page references
  Example: 10,000 instruction
- WSS_i (working set of Process P_i) =
  total number of pages referenced in the most recent Δ (varies in time)
  - ➔if Δ too small will not encompass entire locality.
  - ➔if Δ too large will encompass several localities.
  - ➔if Δ = ∞ ⇒ will encompass entire program.
- $D = \Sigma\ WSS_i \equiv$ total demand frames
- if $D >$ *m(Total number of available frames)* ⇒ Thrashing
- Policy if $D >$ m, then suspend one of the processes.

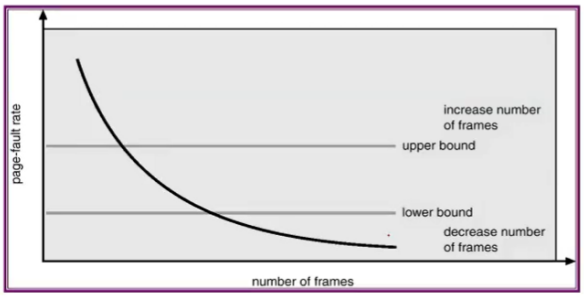
J P Misra

Working-set model



J P Misra

Page-Fault Frequency Scheme



- Establish "acceptable" page-fault rate.
  - ➔If actual rate too low, process loses frame.
  - ➔If actual rate too high, process gains frame.

J P Misra

Other Considerations

- Page Buffering:

➔Maintain a pool of free frames to quickly restart a faulting process

- Prepaging:
  ➔Bring in the complete working set of a swapped out process to avoid initial multiple faults

J

J P Misra

Other Considerations (Cont.)

- **TLB Reach** - The amount of memory accessible from the TLB.
- TLB Reach = (TLB Size) X (Page Size)
- Ideally, the working set of each process is stored in the TLB. Otherwise there is a high degree of page faults.

J

J P Misra

Other Considerations (Cont.)

- Program structure
  - ➔**int A[ ][ ] = new int[1024][1024];**
  - ➔Each row is stored in one page
  - ➔Program 1

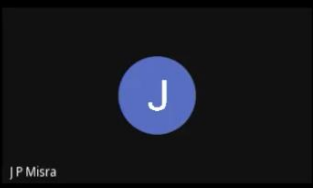
```
for (j = 0; j < 1024; j++)
    for (i = 0; i < 1024; i++)
        A[i,j] = 0;
```

1024 x 1024 page faults

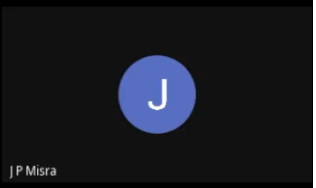
  - ➔Program 2

```
for (i = 0; i < 1024; i++)
    for (j = 0; j < 1024; j++)
        A[i,j] = 0;
```

1024 page faults



Secondary memory



Storage Devices

- Disk storage :
- Solid State storage
- Tape storage
- Hybrid Disk

J

J P Misra

IBM RAMAC 350 disk

- Developed in 1956
- Height 66 inches
- 50 platters of size 24 inches
- Weight : around 1 ton
- Storage capacity : 4MB

J

J P Misra

Attachment

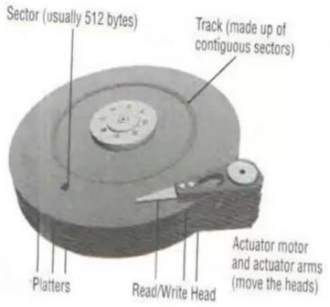
- Drives are attached to computer via **I/O bus**
- USB
- SATA (replacing ATA, PATA, EIDE)
- SCSI
- FC (Fiber Channel) is high-speed serial architecture

J

J P Misra

Anatomy of Disk

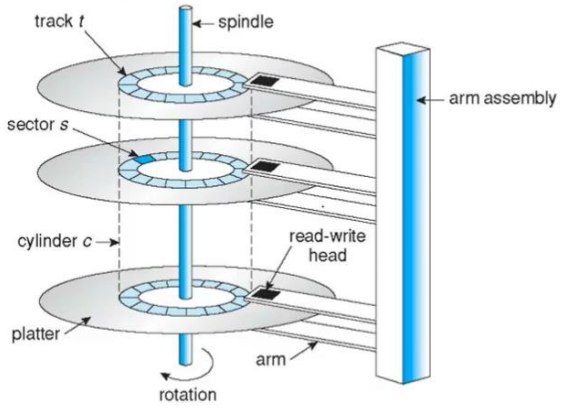
- Major components of disk drives
  - Platters
  - Read write head
  - Actuator assembly
  - Spindle motor



J

J P Misra

Moving-head Disk Mechanism

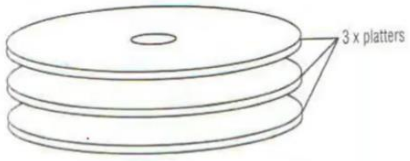


J

J P Misra

Platter

- Are made up of glass or aluminum substrate
- Is coated with magnetic material
- It is rigid, thin, flat , smooth
- All platters are attached to common shaft (spindle)
- Rigidity & smoothness is very important . Any defect could result in head crash

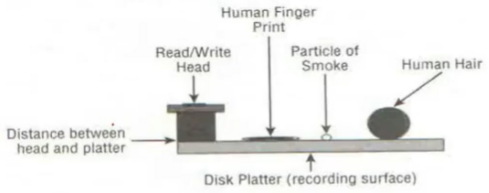


J

J P Misra

Read Write head

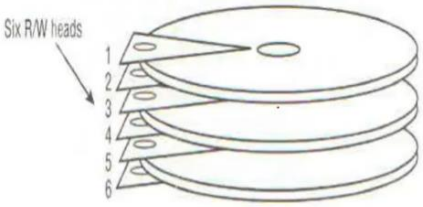
- Head flies above the platter surface
- Attached to actuator assembly
- OS does not know any thing about read write head
- Flying height is measured in micrometers



J P Misra

Head crash

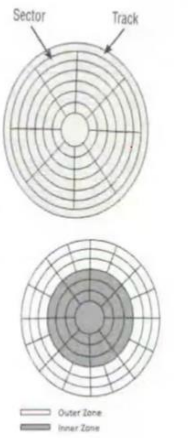
- Read /write head never touch platter. If they touch , it is known as head crash.
- Head crash would always result in complete loss of data .
- Each platter surface has its own read/write head
- Concept of head and recording surface gave rise to CSH addressing



J P Misra

Tracks & Sectors

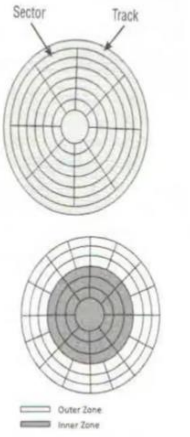
- Surface of every platter is microscopically divided into tracks & sectors
- Sector is the smallest addressable unit of a disk drive
- Sector size is typically 512 bytes or 520 bytes
- SATA (Serial Advance Technology Attachment) have fixed sector size of 512 bytes
- FC (Fiber channel) and serial attached SCSI (SAS) drives can be arbitrarily formatted to different size
  - This is important for implementation of data integrity technology for End to end data protection EDP
- 8 bytes of data added to end of every 512 byte sector
  - This allows errors to be detected



J P Misra

Tracks & Sectors

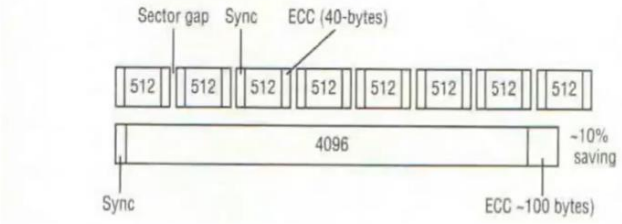
- Size of each sector is getting smaller and smaller towards the center .
- The recording density being same , outer sectors waste space
- Most modern disk implement Zoned Data Recording (ZDR)
- The outer track store and retrieve more data per spin
- If data is read or written to contiguous sector of same or adjacent track we get better performance
- Short stroking to achieve performance
  - Data could be stored only on outermost tracks
  - Reduced capacity , reduced load



J P Misra

Larger sector size trend

- 512 byte sector requires 320 bytes for ECC
- 4K byte requires only 100 byte
- Type of disk where size of sector is fixed and can not be changed is known as Fixed Block Architecture (FBA)



J P Misra

Logical block addressing

- Hides complexity of CSH based addressing
  - Is complex due to ZDR
  - Bad sector handling mechanism
- LBA is implemented in drive controller
- LBA to PBA mapping is required
- OS or volume managers never know the precise location of the data on disk

J P Misra

Controller

- Each disk drive comes with its own controller
- Controller has a processor, memory and it runs firmware
- Controller hides the disk operation complexities
- It maps Physical address to logical address and exposes LBA to OS/ Bios
- Controller also monitors the health of disk
- It also maintains the bad block map
  - Each drive has a number of hidden / spare block which are used to remap bad blocks

J

J P Misra

Common protocol & interfaces

- Serial advance technology attachment (SATA)
- Serial attached SCSI (SAS)
- Nearline SAS (NLSAS)
- Fibre channel (FC)

J

J P Misra

Drive speed

- 5400 RPM
- 7200 RPM
- 10000RPM
- 15000 RPM (240 KM / hr)
- Higher RPM better performance
- Higher RPM lower the capacity
  - 2.5 inch , 900Gb 10K
  - 3.5 inch , 4 TB 7500RPM

J

J P Misra

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 arm 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 ≈ 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.

J

J P Misra

Disk performance

- Seek time (15k 3.4 to 3.9ms) (7.2K 8.5 to 9.5 ms)
- Rotational latency 15k 2ms 7.2k 4.16ms
- Transfer time
- Access time = seek time+ rot delay + transfer time
- IOPS = 1 /(AST+ART) *1000
- eg. 1/(3.2+2)* 1000 = 181

J

J P Misra

Disk Scheduling (Cont.)

- Several algorithms exist to schedule the servicing of disk I/O requests.
- Example: request queue (0-199).

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

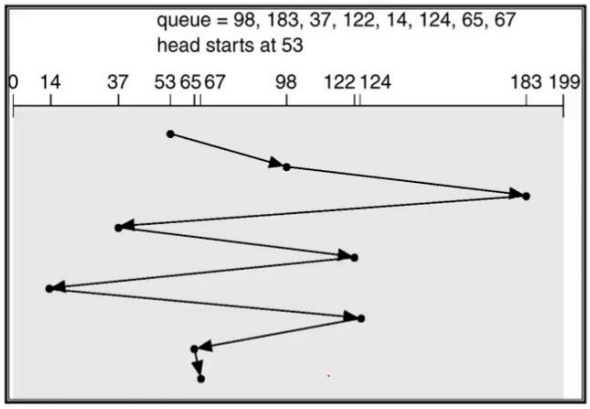
Head pointer 53

J

J P Misra

FCFS

Illustration shows total head movement of 640 cylinders.



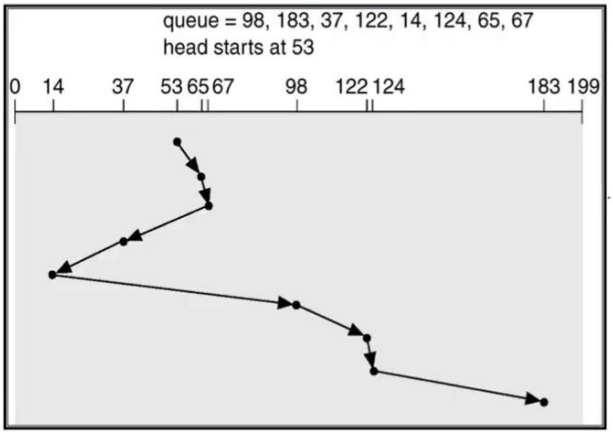
J P Misra

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.

J P Misra

SSTF (Cont.)



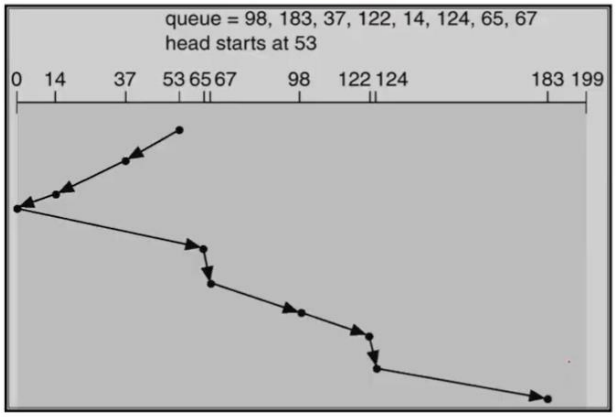
J P Misra

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.

J P Misra

SCAN (Cont.)



J P Misra

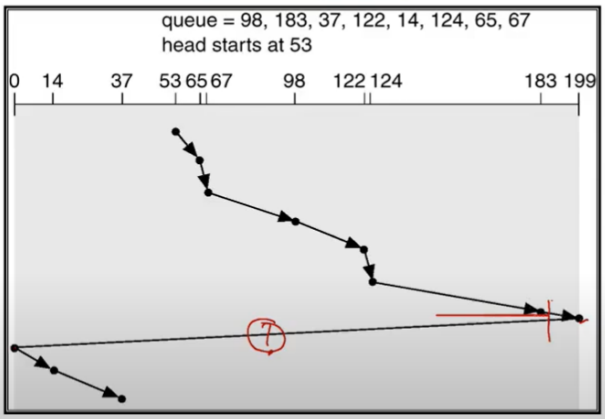
C-SCAN

- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other servicing requests.
- When it reaches the other end, 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.

J P Misra

Operating systems (2020-11-27 at 21:29 GMT-8)

C-SCAN (Cont.)



J

J P Misra

53:33 / 58:46

Operating systems (2020-11-27 at 21:29 GMT-8)

C-LOOK

- Version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately.

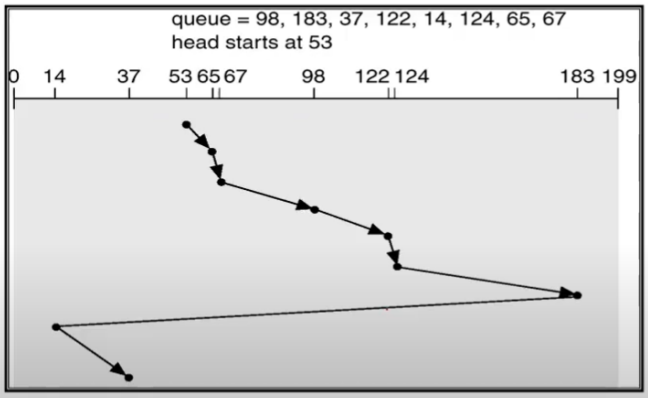
J

J P Misra

53:43 / 58:46

Operating systems (2020-11-27 at 21:29 GMT-8)

C-LOOK (Cont.)



J

J P Misra

53:53 / 58:46

Operating systems (2020-11-27 at 21:29 GMT-8)

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.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

J

J P Misra

54:57 / 58:46

RAID Structure

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