Disk Scheduling

Presented by:

Vaibhav Kumar Gupta

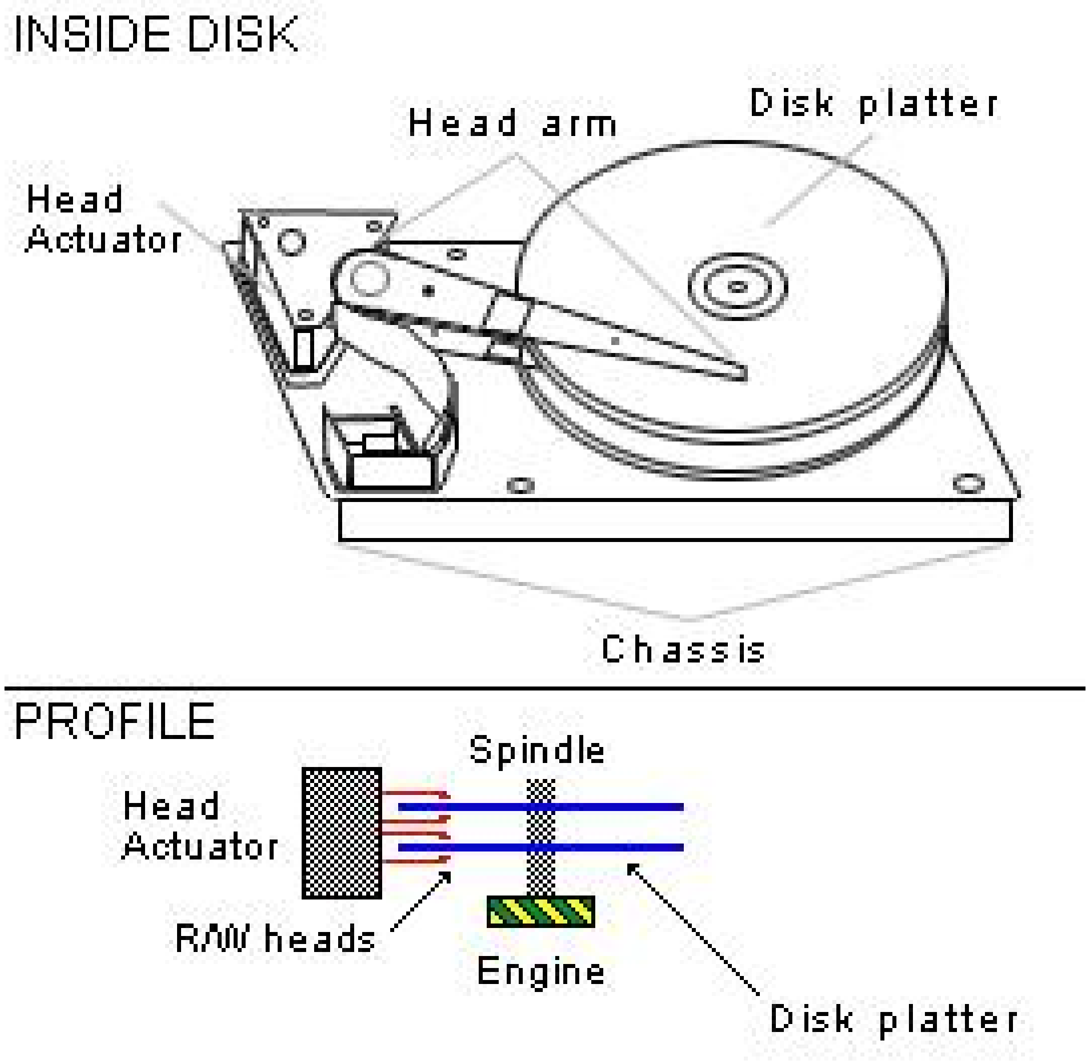
2004EE50416

Overview

- Introduction
- Various Scheduling algorithms
  - FCFS
  - SSTF
  - SCAN Scheduling
  - C-SCAN Scheduling
  - LOOK Scheduling

Disk Scheduling

- What is disk scheduling?
  - Servicing the disk I/O requests
- Why disk Scheduling?
  - Use hardware efficiently
- Includes
  - Fast access time (seek time+ rotational latency)
  - Large disk bandwidth


INSIDE DISK
Head arm
Disk platter
Head Actuator
Chassis
PROFILE
Spindle
Head Actuator
R/W heads
Engine
Disk platter

Disc Scheduling

- I/O request issues a system call to the OS.
  - If desired disk drive or controller is available, request is served immediately.
  - If busy, new request for service will be placed in the queue of pending requests. When one request is completed, the OS has to choose which pending request to service next.

FCFS Scheduling

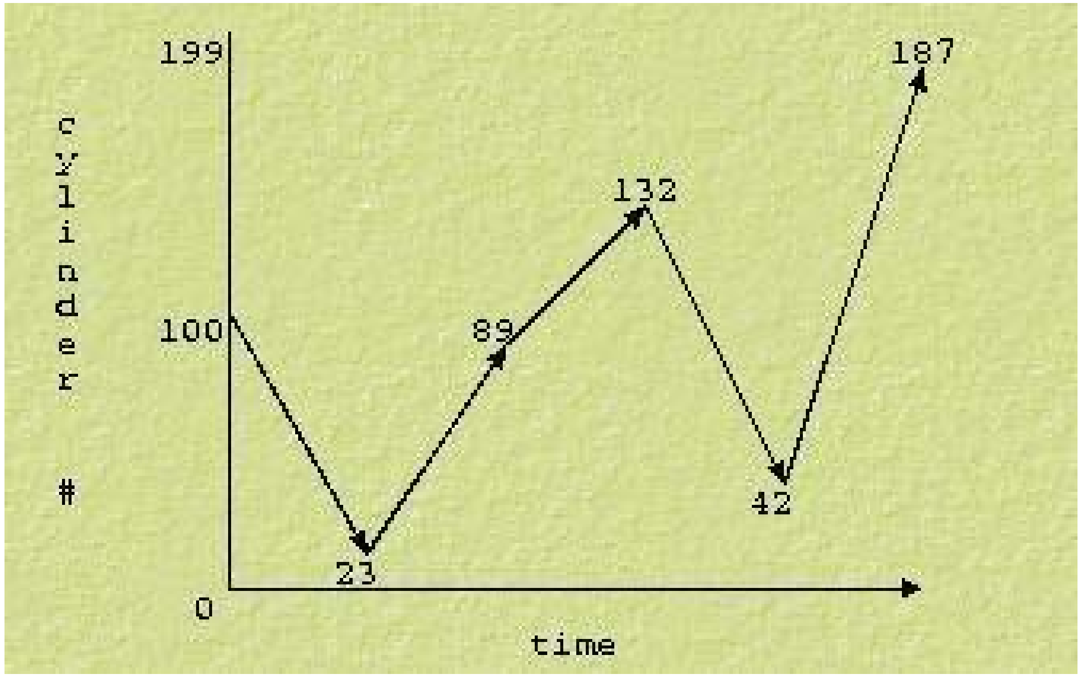
- Simplest, perform operations in order requested
- no reordering of work queue
- no **starvation**: every request is serviced
- Doesn’t provide fastest service
- Ex: a disk queue with requests for I/O to blocks on cylinders

  23, 89, 132, 42, 187

  With disk head initially at 100

FCFS

23, 89, 132, 42, 187



77+66+43+90+145=421

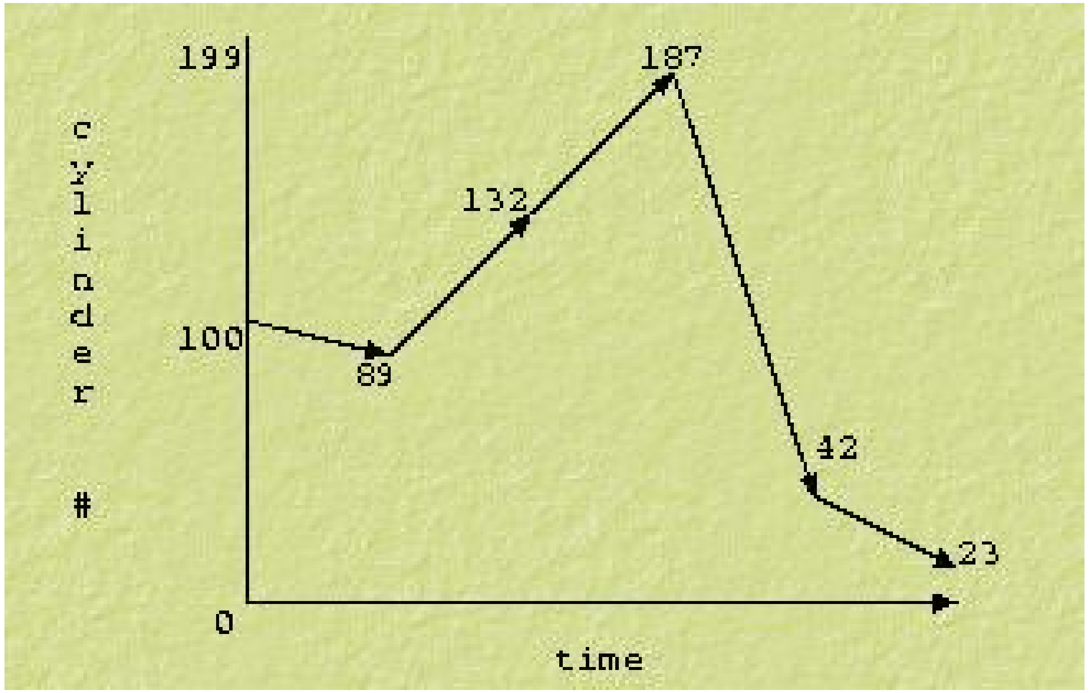
If the requests for cylinders 23 and 42 could be serviced together, total head movement could be decreased substantially.

SSTF Scheduling

- Like SJF, select the disk I/O request that requires the least movement of the disk arm from its current position, regardless of direction
- reduces total seek time compared to FCFS.
- Disadvantages
  - **starvation** is possible; stay in one area of the disk if very busy
  - switching directions slows things down
  - Not the most optimal

SSTF

23, 89, 132, 42, 187



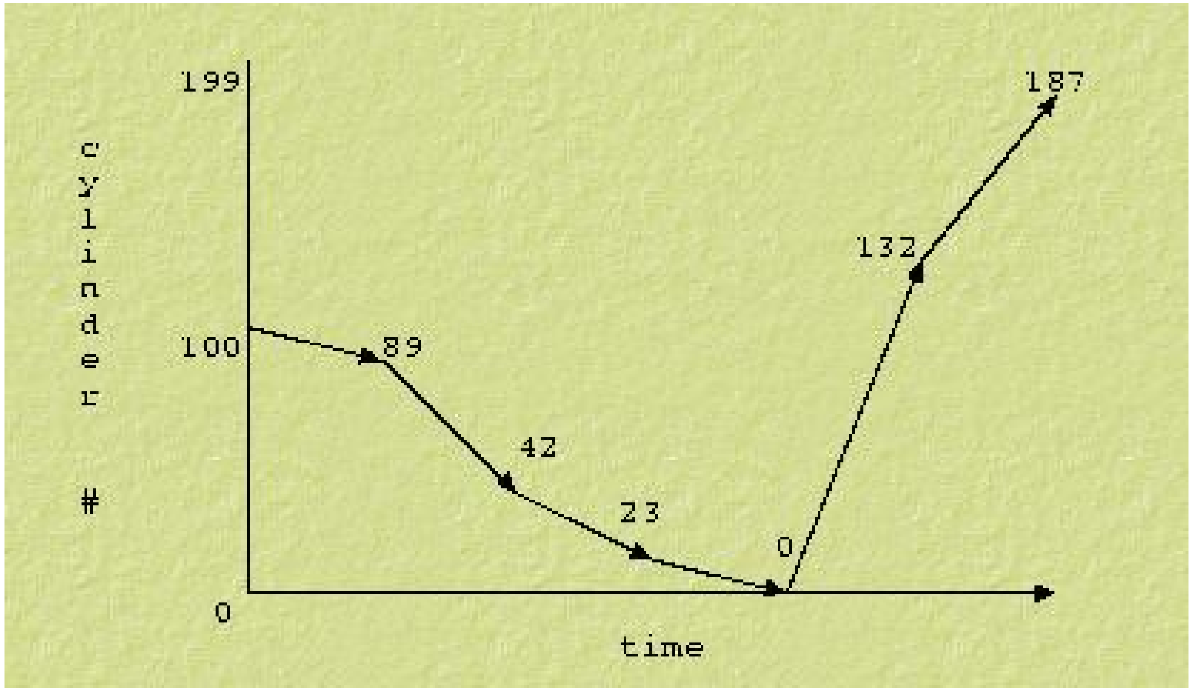
11+43+55+145+19=273

SCAN

- go from the outside to the inside servicing requests and then back from the outside to the inside servicing requests.
- Sometimes called the elevator algorithm.
- Reduces variance compared to SSTF.
- If a request arrives in the queue
  - just in front of the head
  - Just behind

SCAN

23, 89, 132, 42, 187



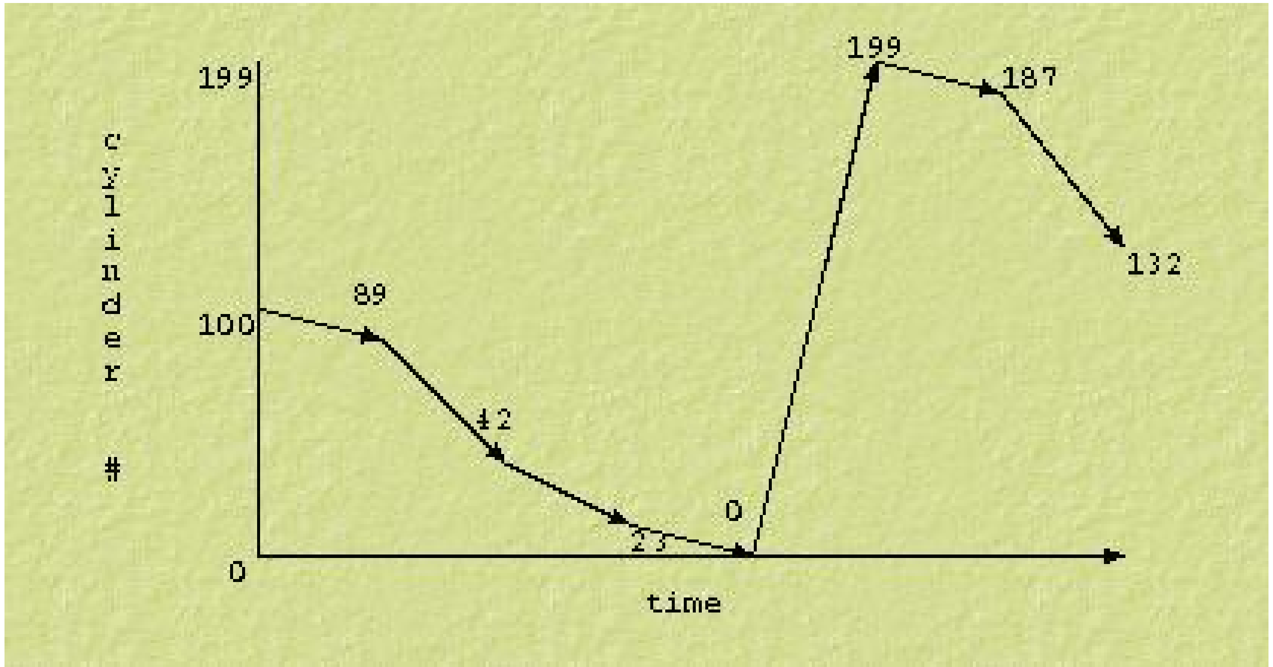
11+47+19+23+132+55=287

C-SCAN

- Circular SCAN
- moves inwards servicing requests until it reaches the innermost cylinder; then jumps to the outside cylinder of the disk without servicing any requests.
- Why C-SCAN?
  - Few requests are in front of the head, since these cylinders have recently been serviced. Hence provides a more uniform wait time.

C-SCAN

23, 89, 132, 42, 187



11+47+19+23+199+12+55=366

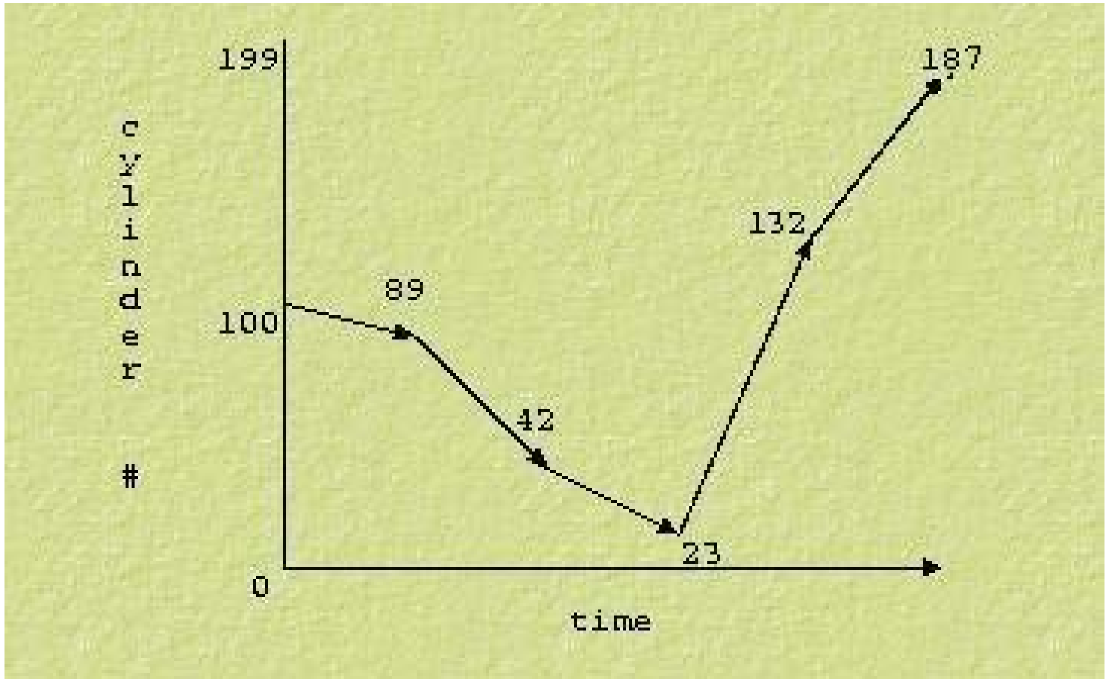
Head movement can be reduced if the request for cylinder 187 is serviced directly after request at 23 without going to the disk 0

LOOK

- like SCAN but stops moving inwards (or outwards) when no more requests in that direction exist

LOOK

23, 89, 132, 42, 187



11+47+19+109+55=241

Compared to SCAN, LOOK saves going from 23 to 0 and then back.

Most efficient for this sequence of requests

Which one to choose?

- Performance depends on number and type of requests.
- SSTF over FCFS.
- SCAN, C-SCAN for systems that place a heavy load on the disk, as they are less likely to cause starvation.
- Default algorithms, SSTF or LOOK

THANK YOU

REFERENCES:

Operating System Principles, Silberschatz, Galvin, Gagne
http://www.dmresearch.net/document/book/Introduction-to-Operating-Systems/notes/io/node8.html
http://hem.passagen.se/communication/ide.html