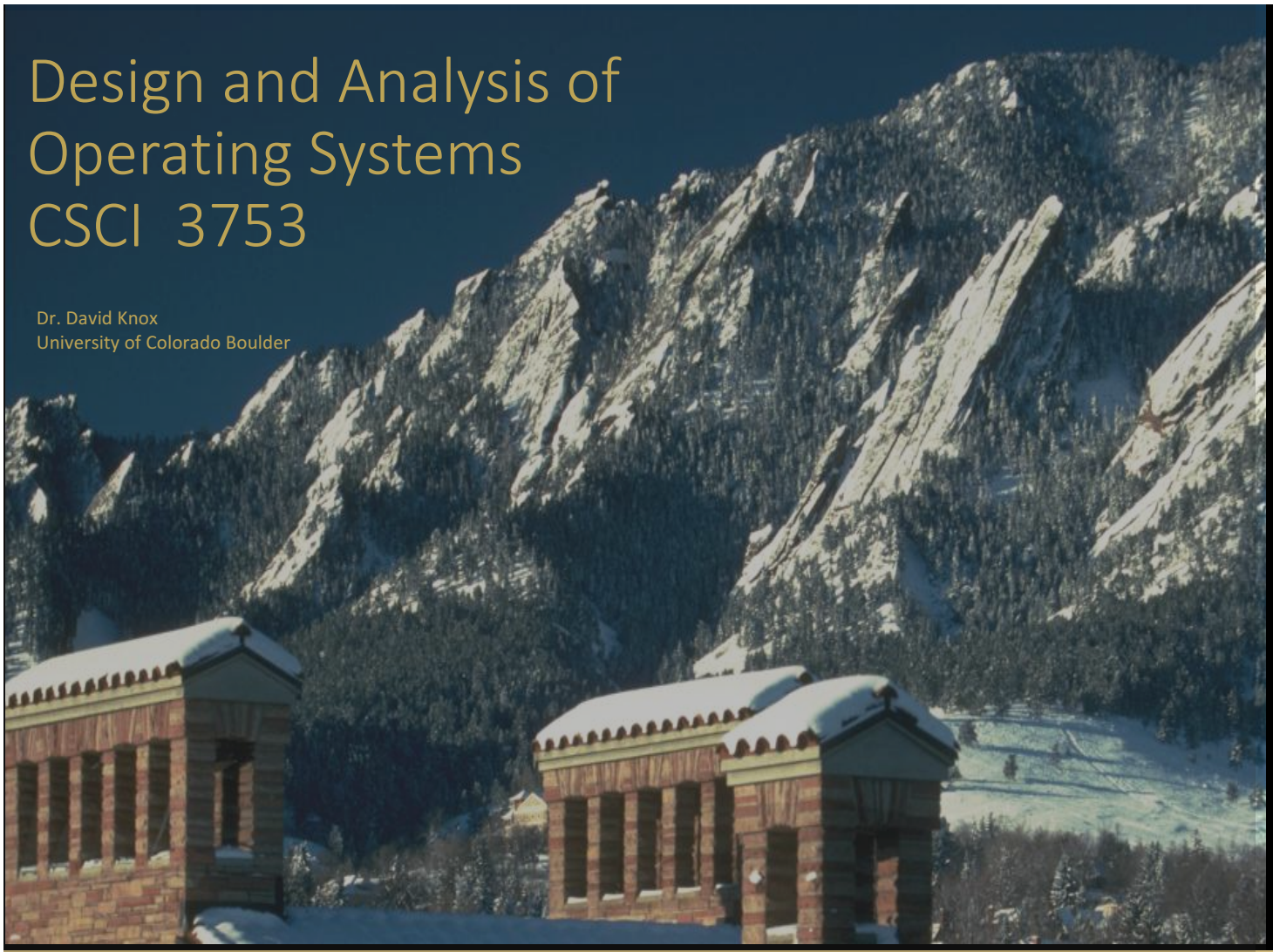
Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of Colorado Boulder

These slides adapted from materials provided by the textbook authors.

Design and Analysis of Operating Systems CSCI 3753



Dr. David Knox
University of Colorado Boulder

Storage Management

Magnetic Media

Magnetic Disk

Magnetic Disk

- Can set a bit to 1 or 0 by applying a strong enough magnetic field to a small region of a magnetic film
  - region retains its magnetic sense even when power is removed
- To read a sequence of bits
  - move the magnetic head over the magnetic media
  - Moving (or rotating) the media under the (static) magnetic head is easier approach
- Supports both random and sequential access
- Array of magnetic disk platters increases bit storage at marginal cost in space and new disk heads

Magnetic Disk

Magnetic Disk

- **Platter**
- **R/W head**
- **Arm**
- **Track**
- **Sector**
- **Cylinder**

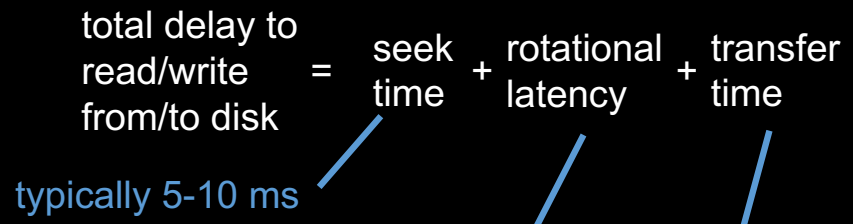
Magnetic Disk
Time to access data

- **Move head into position to read correct track**
  - **5-10 ms**
- **Wait for sector to rotate under head**
  - **1-5 ms depending on rotation speed**
- **Read data from sector on platter (transfer into memory)**
  - **0.001 to 0.1 ms depending on amount of data being transferred**

Magnetic Disk
Time to access data

- **Move head into position to read correct track**
- **Wait for sector to rotate under head**
- **Read data from sector on platter (transfer into memory)**

Disk Access Latency



typically 5-10 ms

typically 1-5 ms, a typical speed (RPM) is ~10000 revolutions per minute.

If on average it takes half a revolution to rotate to the right sector, then time is 0.5/(10000/60) ~= 3 ms

typically 10-100 μs, data transfer rates are GB/s.

Retrieving 10 KB of file data takes 80000/(1 Gb) = .08 ms

- Total disk access delay is often dominated by seek times and rotational latency

Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of Colorado Boulder

These slides adapted from materials provided by the textbook authors.

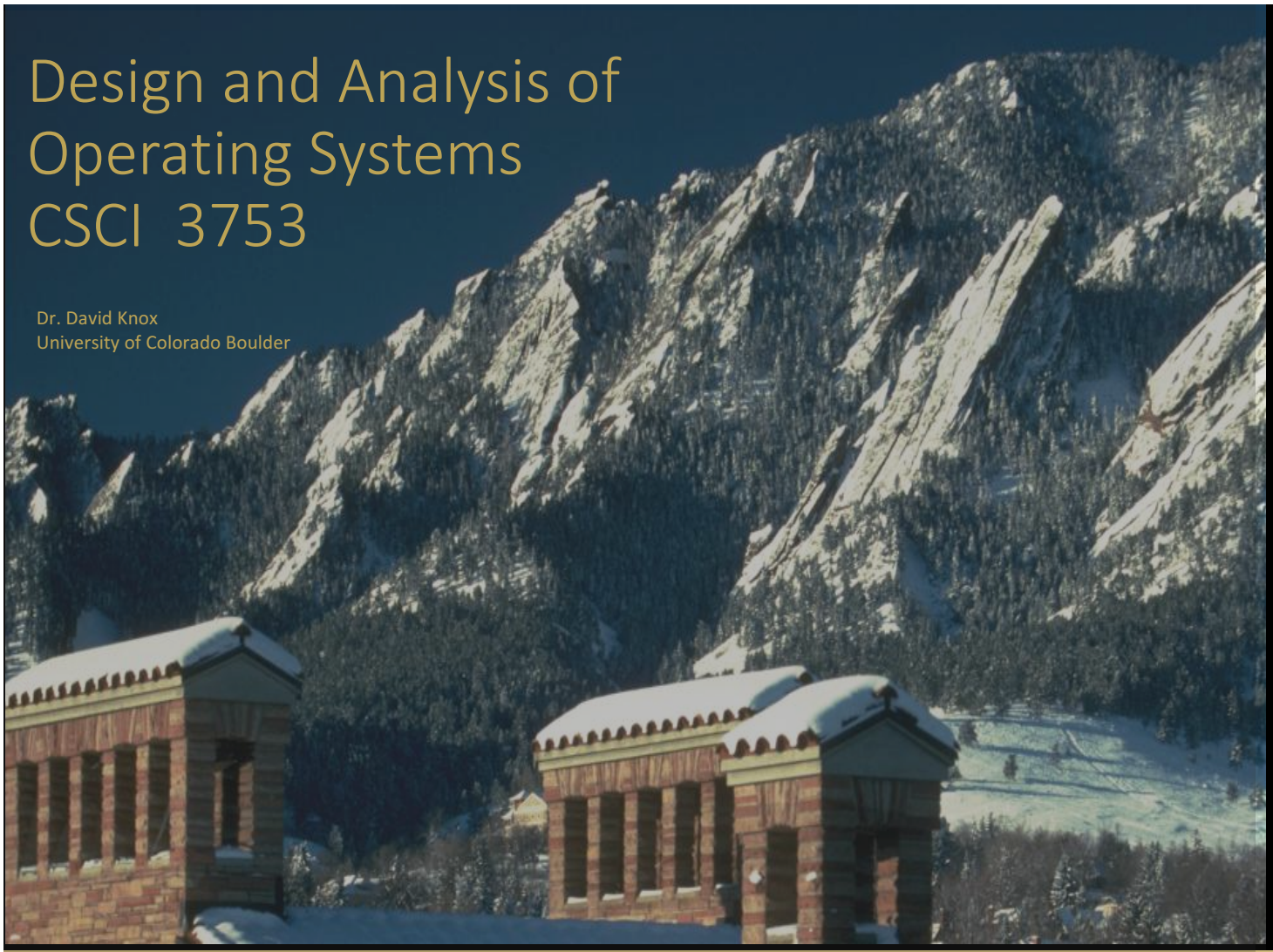
Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of Colorado Boulder

These slides adapted from materials provided by the textbook authors.

Design and Analysis of Operating Systems CSCI 3753



Dr. David Knox
University of Colorado Boulder

Disk Scheduling

Disk Scheduling

- Any technique that can reduce seek times and rotational latency is a big win in terms of improving disk access times
- **Disk Scheduling** is designed to reduce seek times and rotational latency
- Disk operations take time to be processed
  - at any given time there will be multiple requests to access data from different places on the disk
  - these requests are stored in a queue
- The OS can choose to intelligently serve or schedule these requests so as to minimize latency

Our goal: find an algorithm that minimizes seek time…

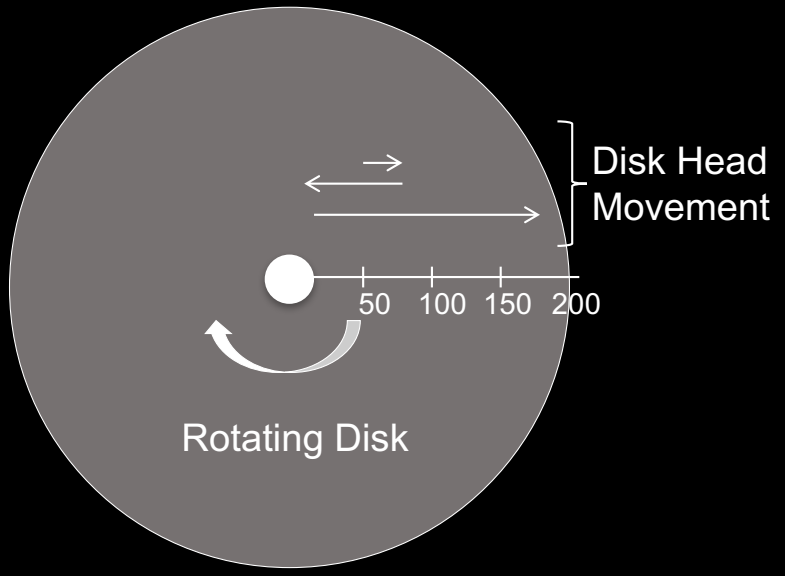
Disk Scheduling

Our goal: find an algorithm that minimizes seek time…

- Suppose we received a series of disk I/O requests for different disk blocks that reside on the following different cylinder/tracks in the following order:

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

- The total number of tracks traversed is used as an evaluation metric of the algorithms



Disk Scheduling

FCFS Disk Scheduling

- Requests would be scheduled in the same order as the order of arrival
- total number of cylinders/tracks traversed by the disk head under FCFS algorithm is ...

- Observation:
  - disk R/W head would move less if 37 and 14 could be serviced together,
  - similarly for 122 and 124

R/W head initially at track 53

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

Disk Scheduling

SSTF Disk Scheduling

- Shortest-Seek-Time-First (SSTF) scheduling selects the next request with the minimum seek time from the current R/W head position
- SSTF services requests in this order

  65, 67, 37 (closer than 98), 14, 98, 122, 124, 183

- Locally optimal, but why not globally optimal?
  - better to move disk head from 53 to 37 then 14 then 65 (this would result in 208 cylinders total)
- Another drawback: starvation
  - while the disk head is positioned at 37, a series of new nearby requests may come into the queue for 39, 35, 38, ...
  - SSTF keeps servicing nearby requests, letting far away requests starve
  - similar to shortest job first process scheduling, which suffers the same problem of starvation

R/W head initially at track 53

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

Disk Scheduling

OPT Disk Scheduling

- OPT looks to minimize the back and forth traversing
- Move the disk head from the current position to the closest edge track, and sweep through once (either innermost to outermost, or outermost to innermost)
- This should minimize the overlap of back and forth and give the minimum # of tracks covered

- OPT services requests as follows:

  53 → 37 → 14 → 65 → 67 → 98 → 122 → 124 → 183

- OPT ignores the initial direction and changes direction

  total # cylinders = 39 down + 169 up = 208 cylinders

- The problem with OPT is that you have to recalculate every time there's a new request for a disk track
- Best approximation to OPT is probably to just scan the disk in one direction, and then change the direction of scanning

R/W head initially at track 53

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

Disk Scheduling

SCAN Disk Scheduling

- Move the read/write head in one direction from innermost to outermost track
- Then reverse direction
  - sweeping across the disk in alternate directions
- SCAN services requests as follows:
  - 53→65→67→98→122→124→183→200→37→14
  - cylinders traversed = 147 up + 186 down = 333
- Advantages:
  - Approximates OPT
  - Starvation free solution
  - Handles dynamic arrival of new requests
  - Simple to implement

R/W head initially at track 53

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

Disk Scheduling

SCAN Disk Scheduling

- Disadvantages:
  - Problem 1: SCAN only approximates OPT
    - In a local time window may not be accurate
    - There is significantly more overlap compared to the OPT service sequence
  - Problem 2: unnecessarily goes to extreme edges of disk even if no requests there
  - Problem 3: it is unfair to the tracks on the edges of the disk (both inside and outside)
    - Writes to the edge tracks take the longest since after SCAN has reversed directions at one edge, the writes on the other edge always wait the longest to be served
    - After two full scans back and forth, middle tracks get serviced twice, edge tracks only once

R/W head initially at track 53

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

Disk Scheduling

C-SCAN Disk Scheduling

- Treat disk as a circular queue of tracks
  - when reaching one edge, move the R/W head all the way back to the other edge
  - continue scanning in the same direction rather than reversing direction
- Circular SCAN, or C-SCAN
- After reaching an edge, writes at the opposite edge get served first
  - this is more fair in service times for all tracks
- C-SCAN services requests as follows:
  - 53→65→67→98→122→124→183→200→1→14→37
  - for a total distance of 384 tracks
- Note how no scanning is done as the disk head is repositioned from 200 all the way down to 1 to ensure circularity

R/W head initially at track 53

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

Disk Scheduling

LOOK Disk Scheduling

- Don't travel to the extreme edge if no requests there
- Look for the furthest request in the current direction
  - only move the disk head that far
  - then reverse direction
- No starvation in this algorithm either
- LOOK would service requests in the following order:
  - 53→65→67→98→122→124→183→37→14
  - total # cylinders traversed = 130 up + 169 down = 299 cylinders

R/W head initially at track 53

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

Disk Scheduling

C-LOOK Disk Scheduling

- Improve LOOK by making it more fair
  - Circular LOOK
- after moving disk head to furthest request in current direction
  - move disk head directly all the way back to the further request in the other direction
  - continue scanning in the same direction
- C-LOOK would service requests in the following order:
  - 53→65→67→98→122→124→183→14→37
  - total # cylinders traversed = 130 up + 169 down + 23 up = 322 cylinders

R/W head initially at track 53

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

Disk Scheduling

Disk Scheduling in Practice

- In the previous examples, the # of cylinders traversed by SCAN, LOOK, and C-LOOK are much lower if the initial direction was down
- We can't keep changing the direction
  - Can cause starvation/unfairness
  - Preserving directionality of the scan is important for fairness, but we sacrifice some optimality/performance
- Either SSTF or LOOK are reasonable choices as default disk scheduling algorithms
- These algorithms for reducing seek times are typically embedded in the disk controller today
- There are other algorithms for reducing rotational latency, also embedded in the disk controller

R/W head initially at track 53

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

Disk Scheduling

File Layout and Disk Scheduling

- Block allocation schemes can effect access times
  - When a file's data is spread across the disk, it increasing seek times
  - Contiguous allocation minimizes seek times
  - Select free blocks near adjacent file data
- Opening a file for the first time requires searching the disk for the directory, file description header, and file data
  - these are all spread out across the disk
  - seek times are large
- Could store directories in the middle tracks, so at most half the disk is traversed to find the file header and data
- Could store the file header near the file data, or near the directory

R/W head initially at track 53

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

Disk Scheduling

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

Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Design and Analysis of Operating Systems CSCI 3753

Dr. David Knox
University of Colorado Boulder

These slides adapted from materials provided by the textbook authors.