Overview

- NewSQL
  - Introduction & Main Memory Databases
  - Solid State Disk and Databases
  - Transactions on Multicores
- NoSQL
  - Graph Databases
  - Document Databases
  - Document & Graph Databases Tutorial
- One assessed coursework

Database Storage Layer

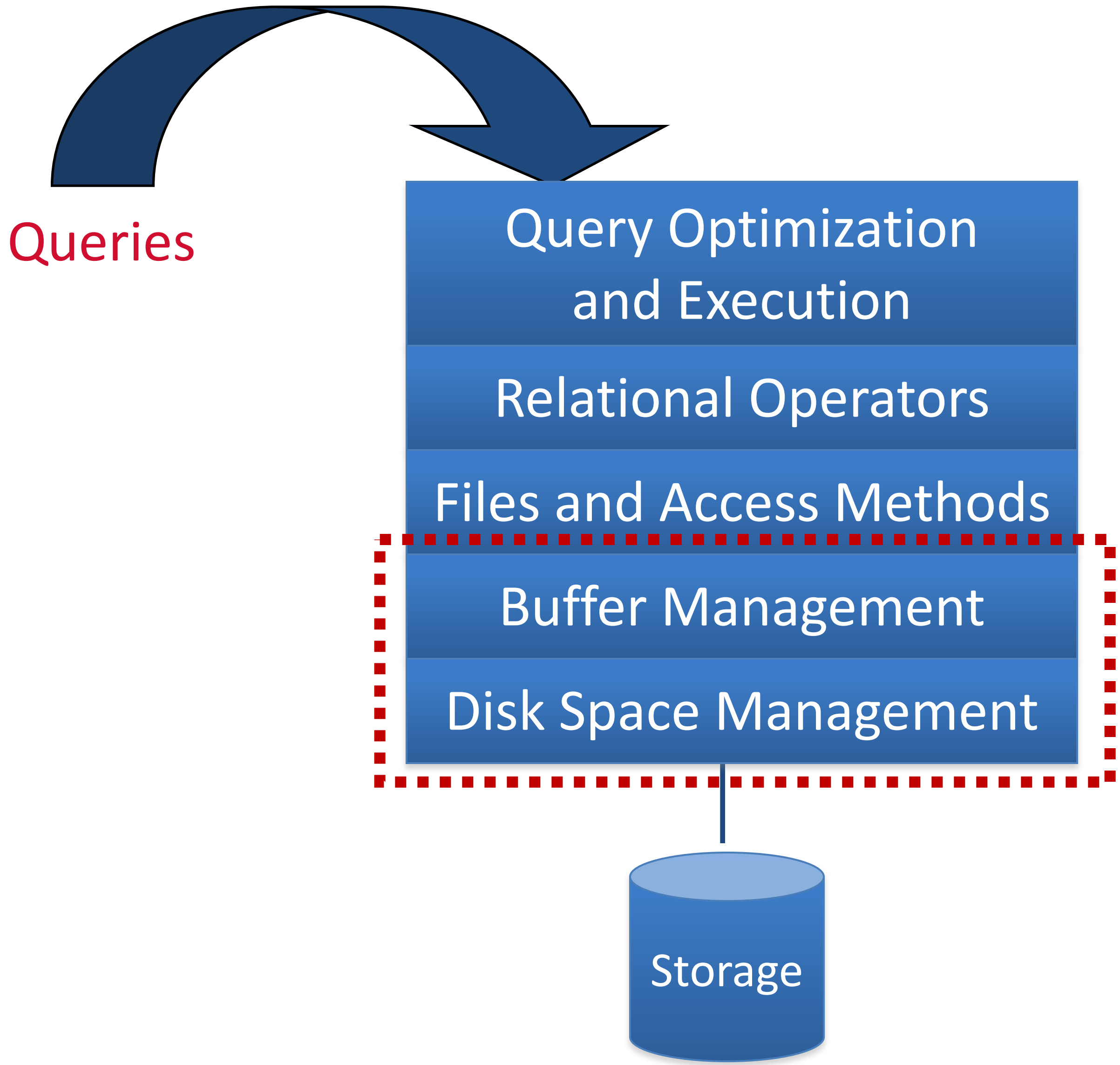
Thomas Heinis
t.heinis@imperial.ac.uk
Scale Lab - scale.doc.ic.ac.uk

SCALE LAB
Imperial College London

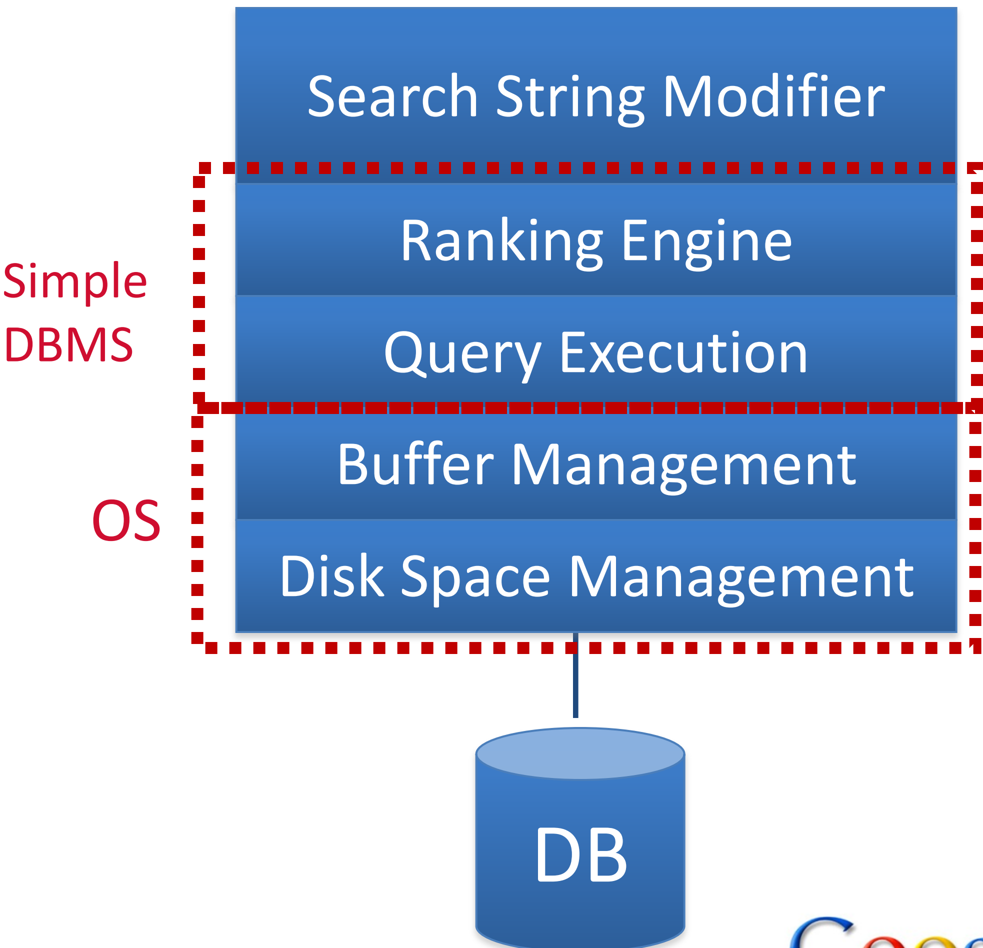
The Storage Layer

- DBMS layers and storage hierarchy
- Disks

DBMS Layers


Queries
Query Optimization
and Execution
Relational Operators
Files and Access Methods
Buffer Management
Disk Space Management
Storage

A simple search engine



- Simpler system than DBMS
  - Uses OS files for storage
  - One hardwired query
- Typically no concurrency/ recovery
  - Read-mostly, in batches
  - No updates to recover
  - OS a reasonable choice
- Smarts: text tricks
  - Search string modifier (synonyms)
  - Ranking Engine (sort results)
  - no semantics

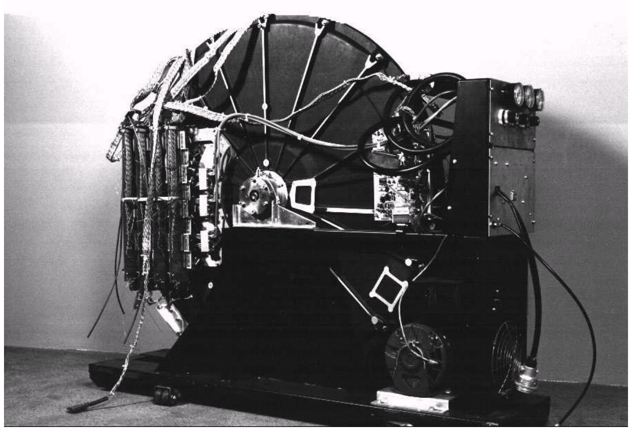
Google bing much more complex

Why not OS?

- Layers of abstraction are good … but:
  - Unfortunately, OS often gets in the way of DBMS

- DBMS needs to do things "its own way"
  - Specialized prefetching
  - Control over buffer replacement policy
    - LRU not always best (sometimes worst!!)
  - Control over thread/process scheduling
    - "Convoy problem"
      - Arises when OS scheduling conflicts with DBMS locking
  - Control over flushing data to disk
    - WAL protocol requires flushing log entries to disk

Disks and Files



- DBMS stores information on disks.
  - In an electronic world, disks are a mechanical anachronism!
- This has major implications for DBMS design!
  - READ: transfer data from disk to main memory (RAM).
  - WRITE: transfer data from RAM to disk.
  - Both are high-cost operations, memory operations, so must be planned carefully!

Why Not Store It All in Main Memory?

- *Costs too high*
  - High-end Databases today in the Petabyte range.
  - ~ 60% of the cost of a production system is in the disks.
- *Main memory is volatile.* We want data to be saved between runs. (Obviously!)

- *But, main-memory database systems do exist!*
  - Smaller size, performance optimized
  - Volatility is ok for some applications

What about Flash?

- Flash chips used for >20 years

- Flash evolved

  - USB keys
  - Storage in mobile devices
  - Consumer and enterprise flash disks (SSD)

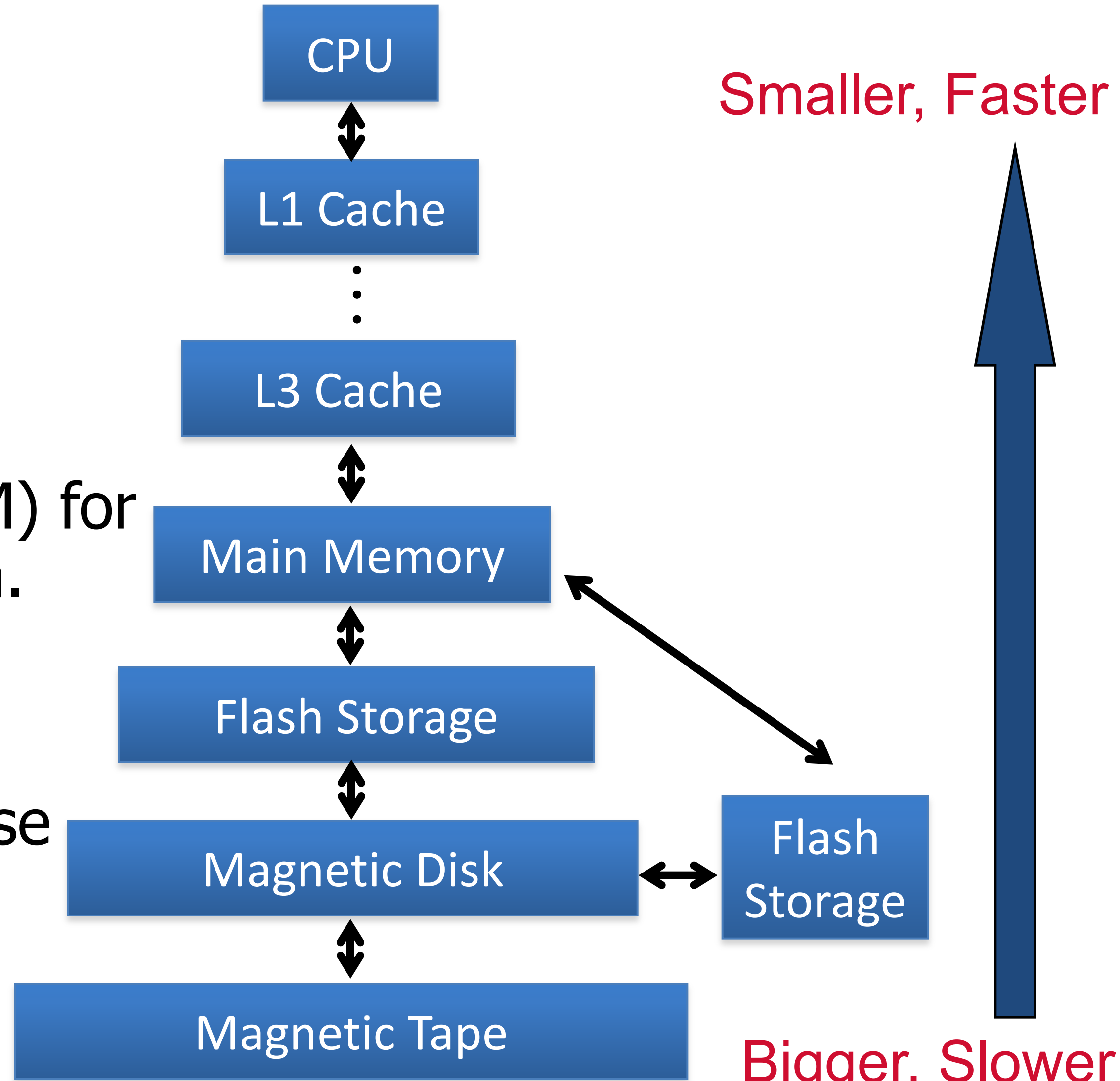
- Flash in a DBMS
  - Main storage
  - Accelerator/enabler (specialized cache, logging device)

The Storage Hierarchy

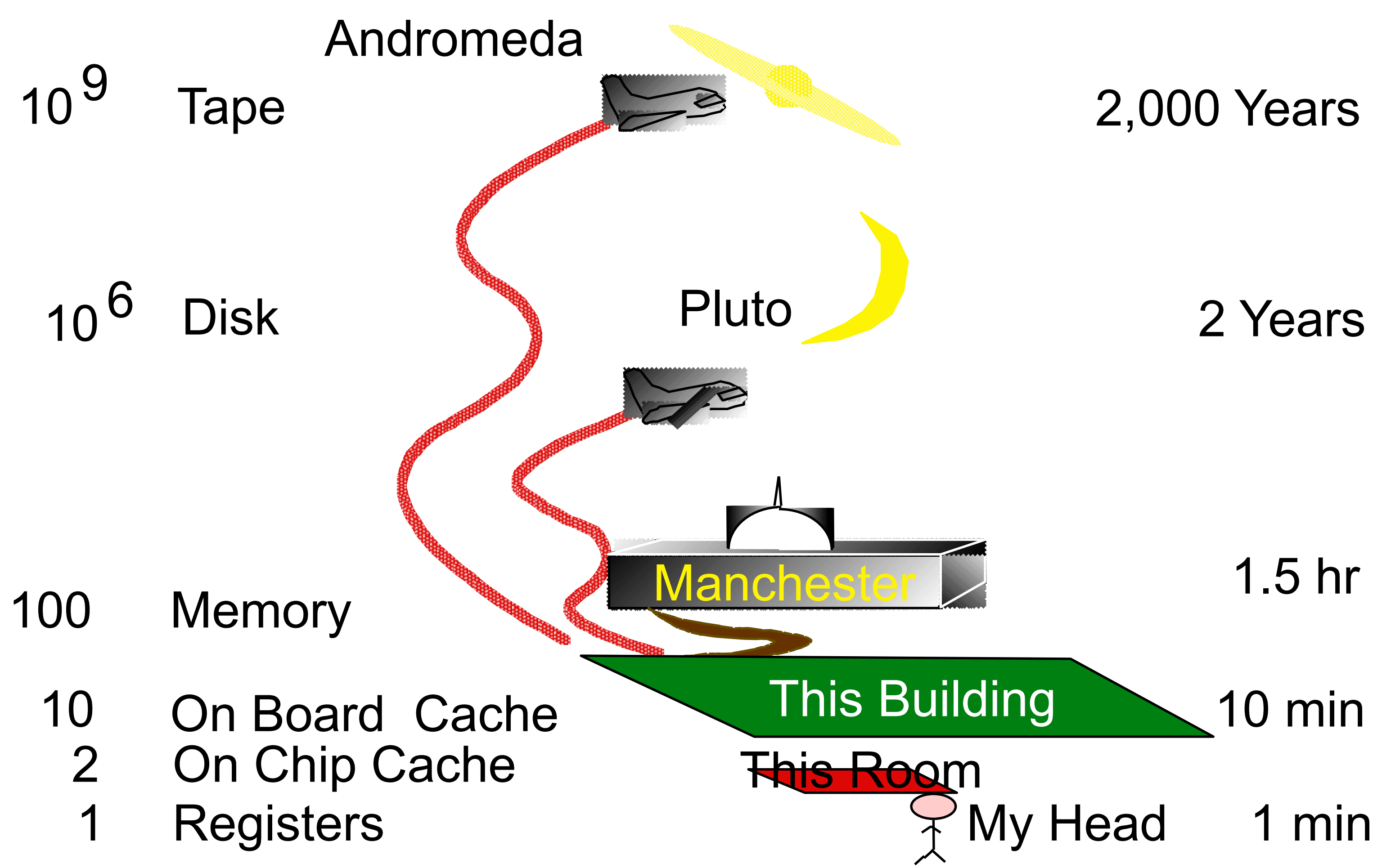
Main memory (RAM) for currently used data.

Disk for the main database (secondary storage).

Tapes for archival storage (tertiary storage).



Jim Gray's Storage Latency Analogy: How Far Away is the Data?



The Storage Layer

- DBMS layers and storage hierarchy

- Disks

Disks

- Secondary storage device of choice.
- Main advantage over tapes: *random access* vs. *sequential*.
- Data is stored and retrieved in units called *disk blocks* or *pages*.
- Unlike RAM, time to retrieve a disk page varies depending on location on disk.
  - Therefore, relative placement of pages on disk has major impact on DBMS performance!

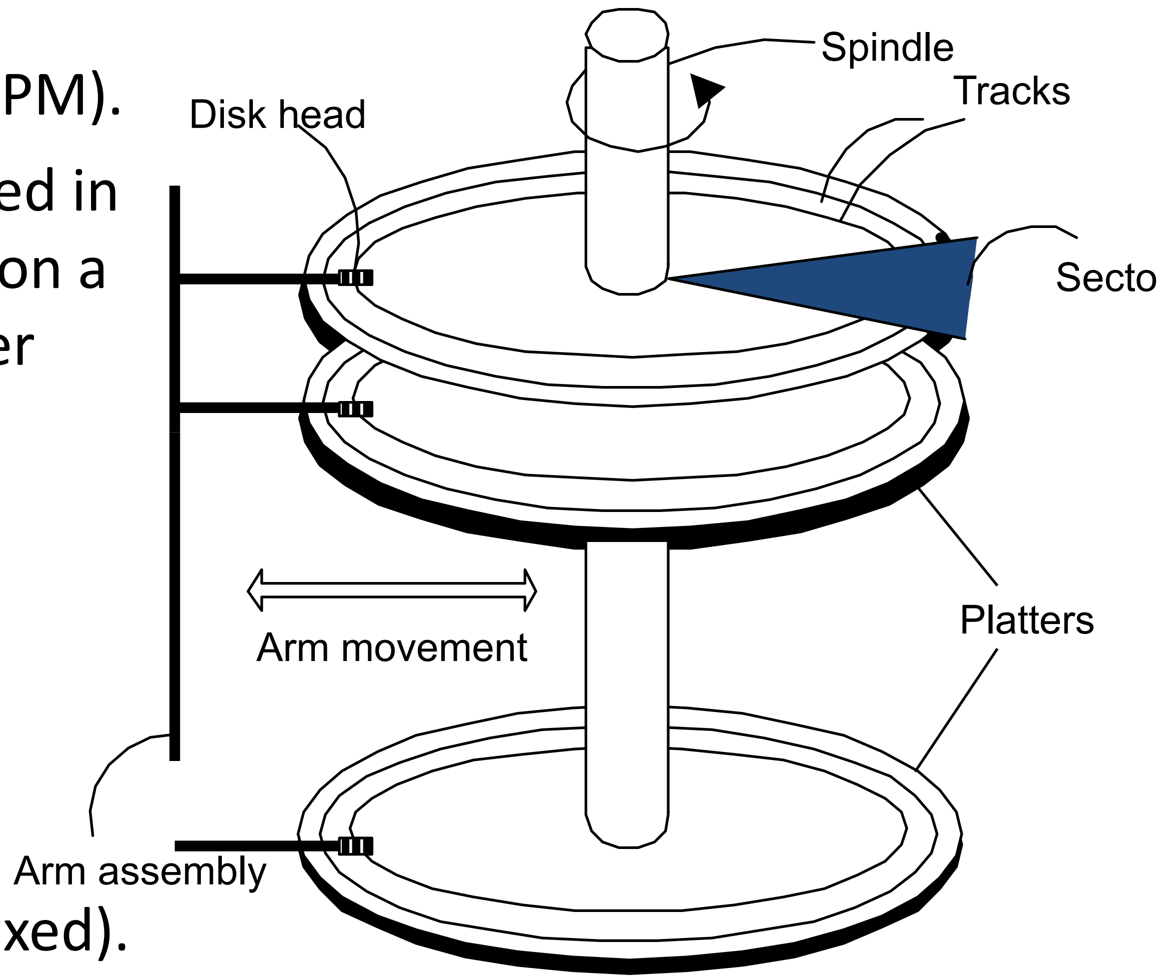
Anatomy of a Disk

The platters spin (5-15 kRPM).

The arm assembly is moved in or out to position a head on a desired track. Tracks under heads make a *cylinder* (imaginary!).

Only one head reads/writes at any one time.

- *Block size* is a multiple of *sector size* (which is fixed).
- Newer disks have several "zones", with more data on outer tracks.

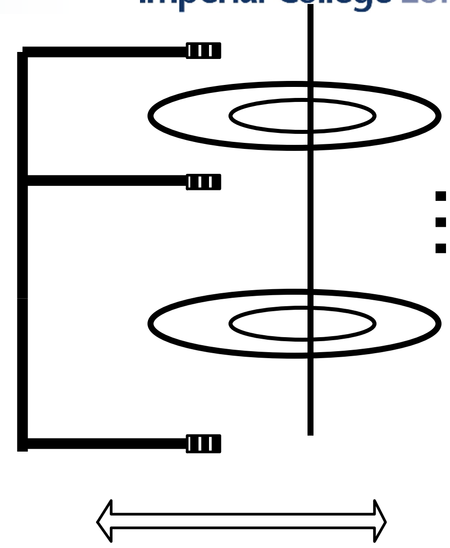


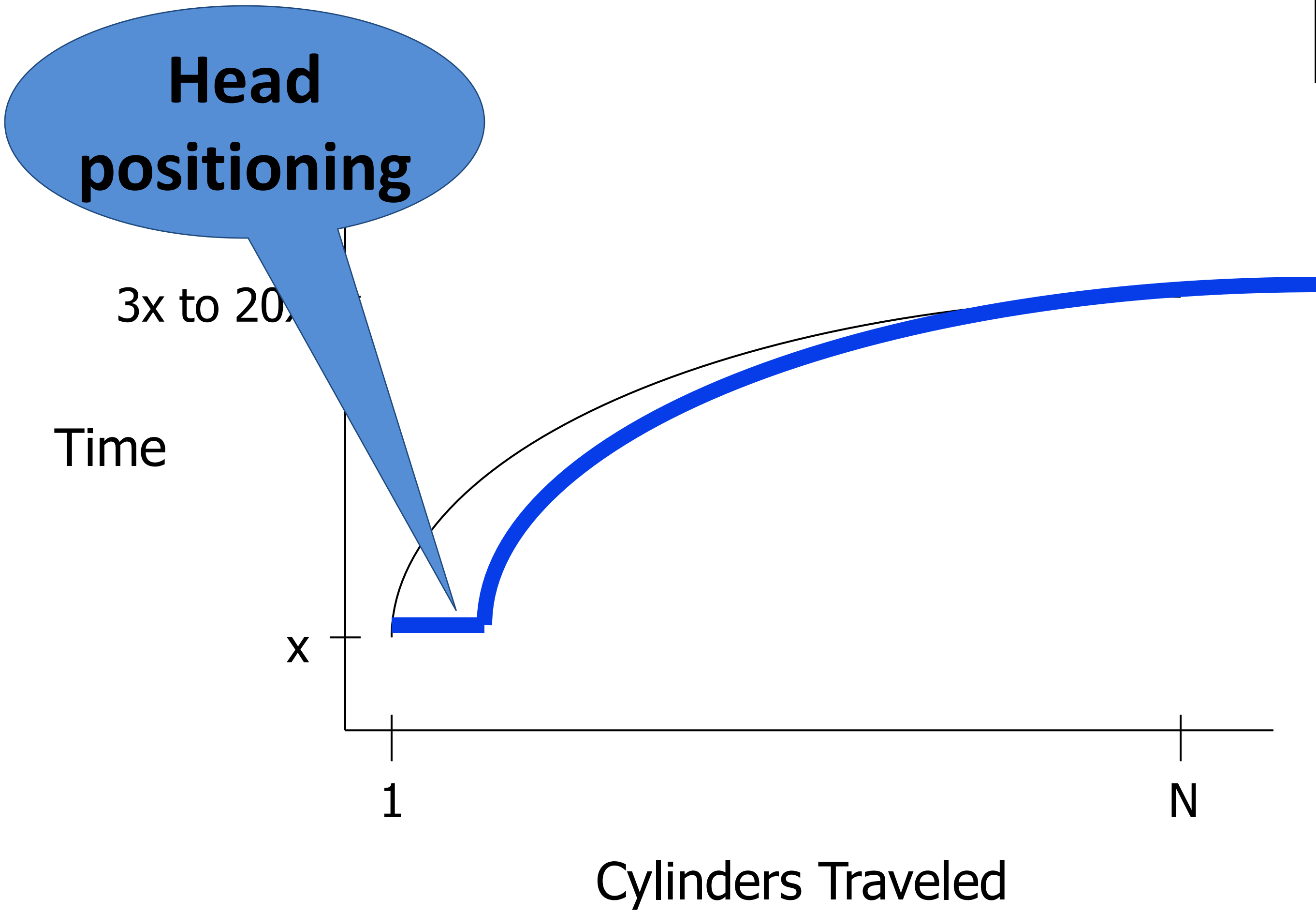
Accessing a Disk Page

Time to access (read/write) a disk block:

- *seek time* (moving arms to position disk head on track)
- *rotational delay* (waiting for block to rotate under head)
- *transfer time* (actually moving data to/from disk surface)

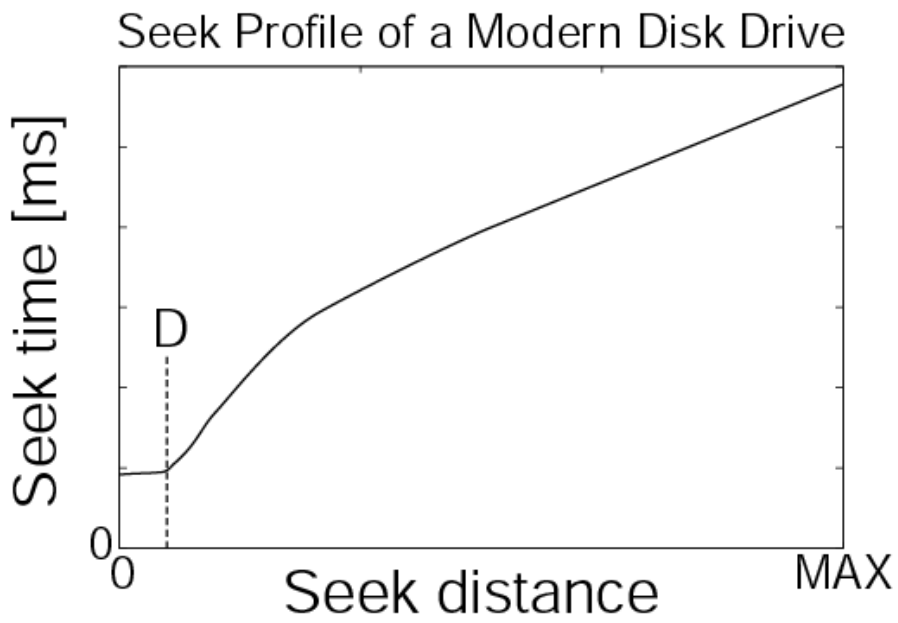
Seek Time


Arm movement


Head positioning
3x to 20
Time
x
1
N
Cylinders Traveled

Seeking in Modern Disks

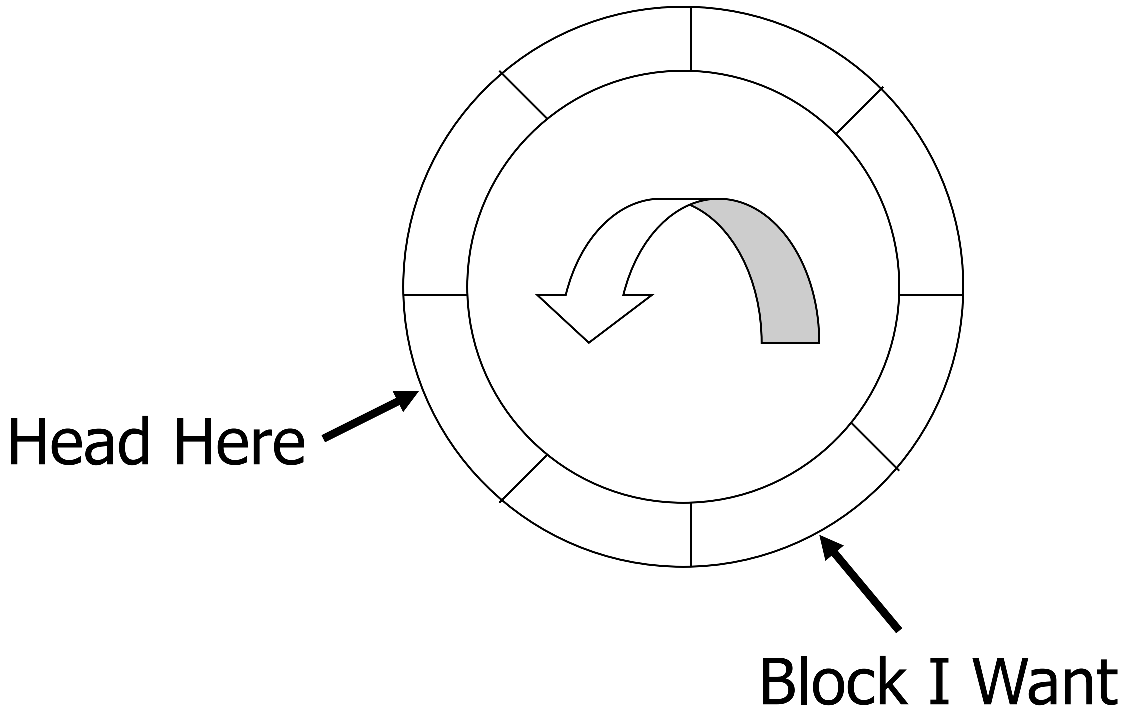
Seek time discontinuity



Short seeks are dominated by "settle time"

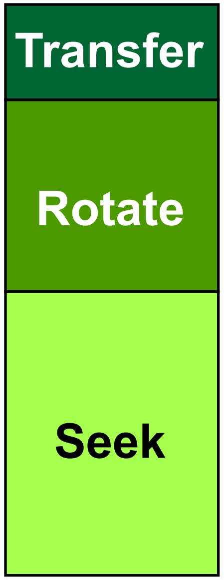
- Move to one of many nearby tracks within settle time
- D is on the order of tens to hundreds
- D gets larger with increase of disk track density

Rotational Delay



Seek Time & Rotational Delay Dominate

- Seek time varies from about 1 to 20 ms
- Rotational delay varies from 0 to 10 ms
- Transfer rate is < 1ms per 4KB page



- Key to lower I/O cost:
  reduce seek/rotation delays!

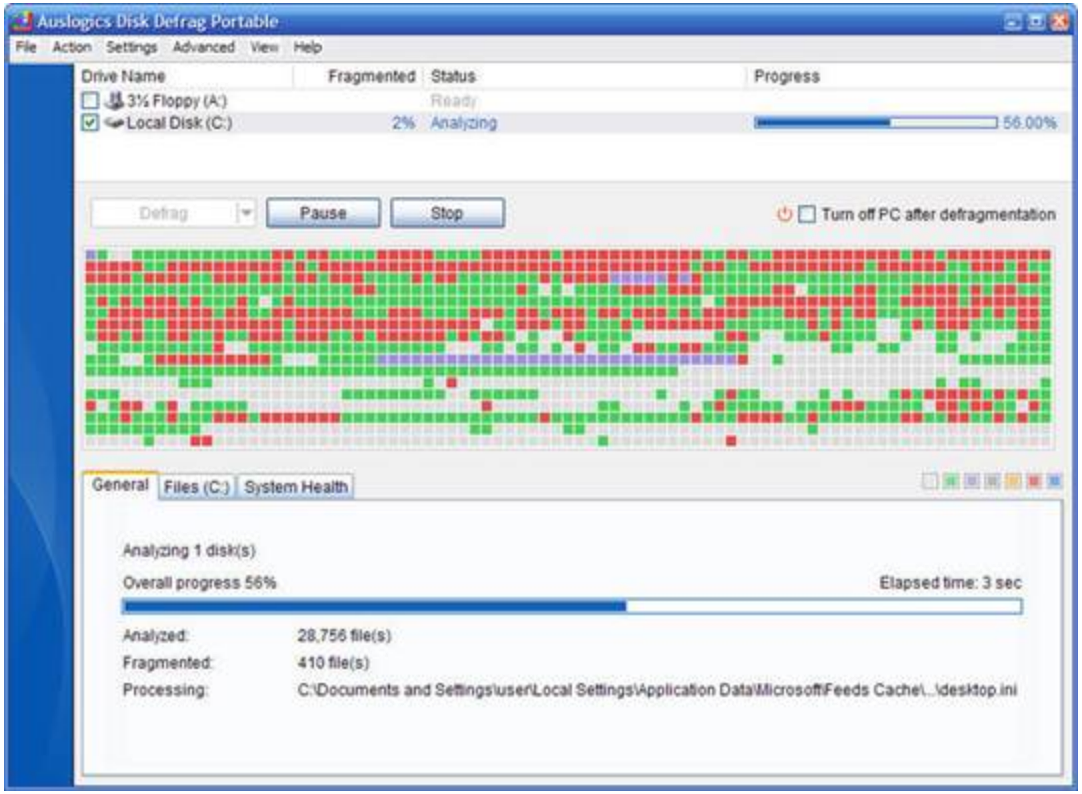
- Also note: For shared disks most time spent waiting in queue for access to arm/controller

Arranging Pages on Disk

- *"Next"* block concept:
  - blocks on same track, followed by
  - blocks on same cylinder, followed by
  - blocks on adjacent cylinder
- Blocks in a file should be arranged sequentially on disk (by "next") to minimize seek and rotational delay.

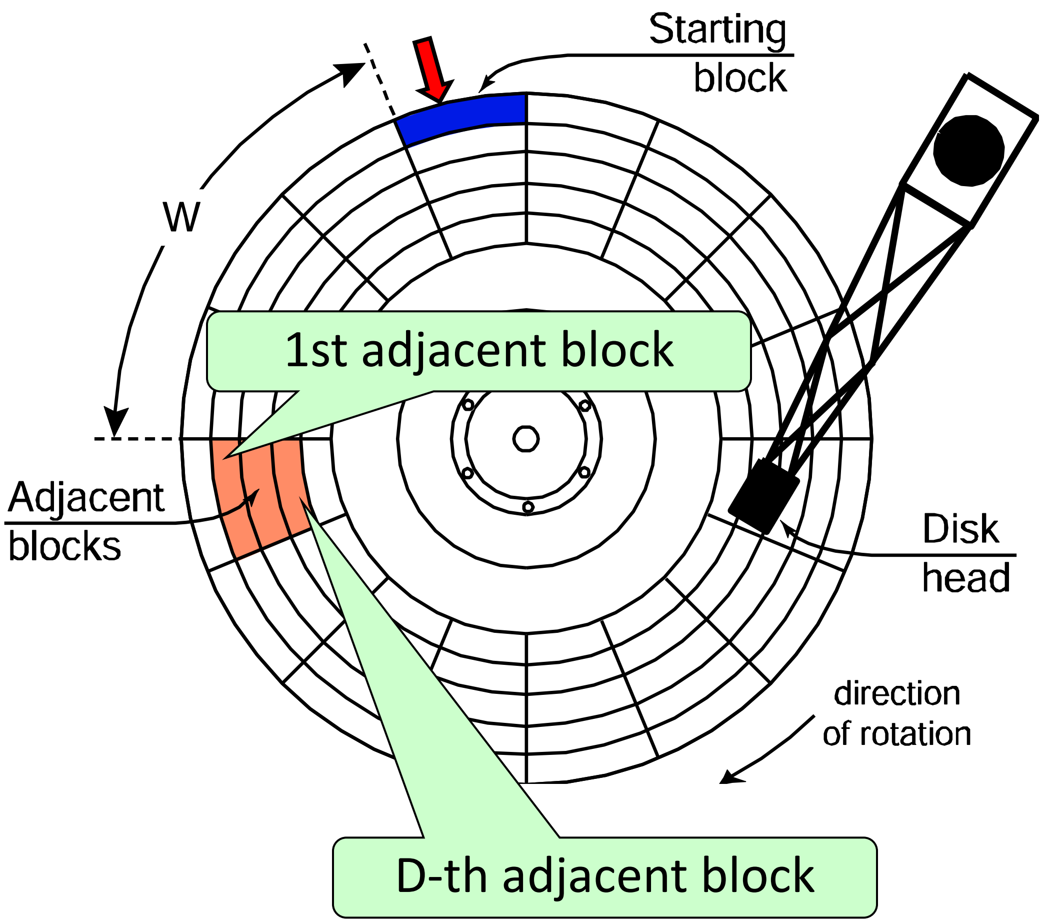
- An important optimization: pre-fetching

Remember Defrag



Define adjacent blocks

- Access incurs settle time only
- Equidistant wrt access time from starting block



D: # of adjacent blocks

W: degree disk will rotate during settle time

Disk block has more than one neighbor

Rules of thumb…

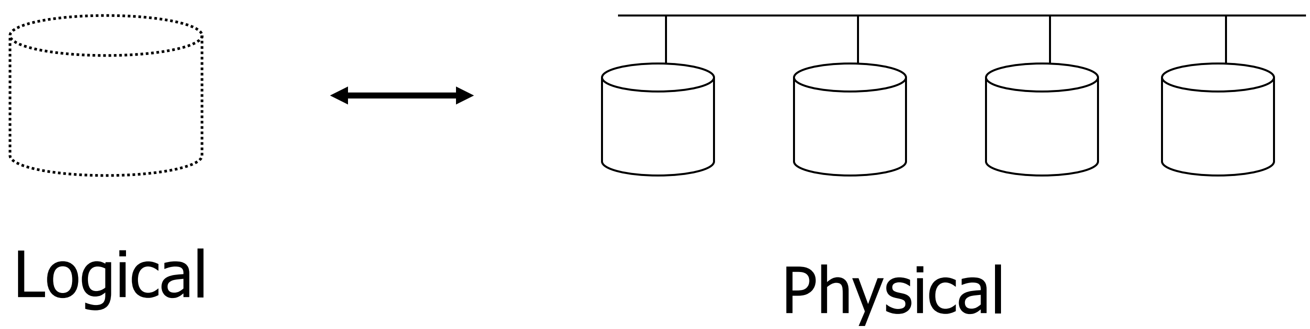
1. Memory access much faster than disk I/O (~ 1000x)

2. “Sequential” I/O faster than “random” I/O (~ 10x)

Disk Space Management

- Lowest layer of DBMS software manages space on disk
- Higher levels call upon this layer to:
  - allocate/de-allocate a page
  - read/write a page
- Best if a request for a *sequence* of pages is satisfied by pages stored sequentially on disk! Higher levels don't need to know if/how this is done, or how free space is managed.

Disk Arrays: RAID



- Benefits:
  - Higher throughput (via data "striping")
  - Longer MTTF (via redundancy)

Summary

Key to store data on disk is:

1. Store data together if it is queried together
2. Avoid random access and use sequential access where possible: use cost model for it
3. Unit to optimize for is disk page: align data structures for page size