

CS 162 Spring 2007 Midterm 2 Review

*Thomas Kho
tkho@eecs*

2007-04-09

*Note: some of the illustrations in this presentation are by Silberschatz, Galvin and Gagne ©2002
Based on the CS 162 Spring 2005 Midterm 2 Review by Karl Chen and Adrian Mettler*

Metacomments

- Midterm 2
 - Wednesday, 2007-04-11 19:00 in 306 Soda
- Midterms are cumulative
- Do previous midterms by Smith

Available Study Materials

- Professor's Lecture Notes
- Student Lecture Notes
- Discussion Notes
- Sample Midterm in Reader
- Reader, Textbook

Agenda

- Caches & TLBs
- Paging
- I/O, Disks
- File systems, directory structure
- RAID
- Networks

- Sample problems

Caches and TLBs - motivations

- What is the primary motivation behind using a cache?
 - Principle of locality
 - Temporal locality
 - Spatial locality
 - Difference in access time and capacity in the memory hierarchy

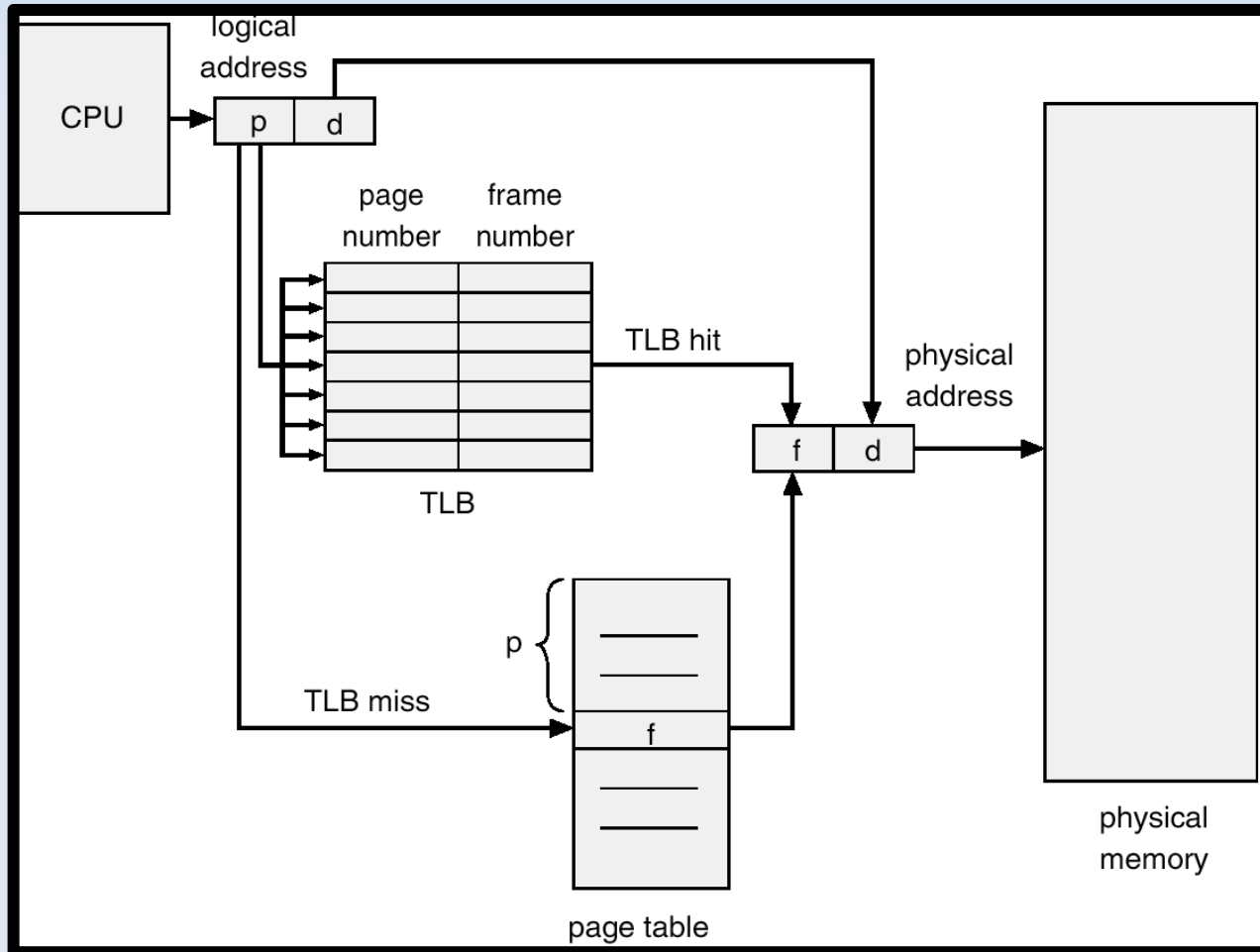
Searching a cache

- What are the different methods of organizing/searching a cache in hardware?
 - What is direct mapped?
 - What is N-way set associative?
 - What is fully associative?
 - Advantages and disadvantages of each?

Caches and TLBs, cont. . .

- What is a TLB?
 - Why do we use TLBs?
- What happens to the TLB on a context switch?
- What does using a TLB add to handling page faults?
- What is the effective access time of a TLB, a memory reference?
 - E.g. how is it calculated?

Caches and TLBs, cont. . .



Paging

- What happens when you write to a memory location that's swapped out?
 - TLB lookup
 - TLB fault
 - Page table lookup
 - Page fault
 - Context switch

Page Faults, cont. . .

- What does the OS do after handling a page fault?
- How many page faults can occur in one instruction?
- Managing pages
 - Page fetch algorithm: When to fetch pages into memory
 - Page replacement algorithm: which pages should be replaced
 - Page placement algorithm: where to put the page

Page Fetch Algorithms

- **Demand Paging**
 - What is this?
 - Why is it non-optimal?
- **Request Paging**
 - What is this?
 - Why is this somewhat non-feasible?
- **Prefetching**
 - What is this?
 - What's the problem with it?
- **Overlays – not using virtual memory**
- **Be able to work each of these into any flowchart/description of paging!**

Page Replacement Algorithms

- Random
- MIN/OPT
- FIFO
 - Advantages?
 - Limitations of not being a stack algorithm (Belady's Anomaly)?
- LRU
 - Limitations in implementing it?
 - Why can the opposite (MRU) be better?
- FINUFO/Clock
 - What is this an approximation of?
 - How does it work?

Page Table Entry fields

- What is the Valid bit in a PTE for?
- What is the Reference/Used bit for?
- What is the Dirty bit for?

Pinning

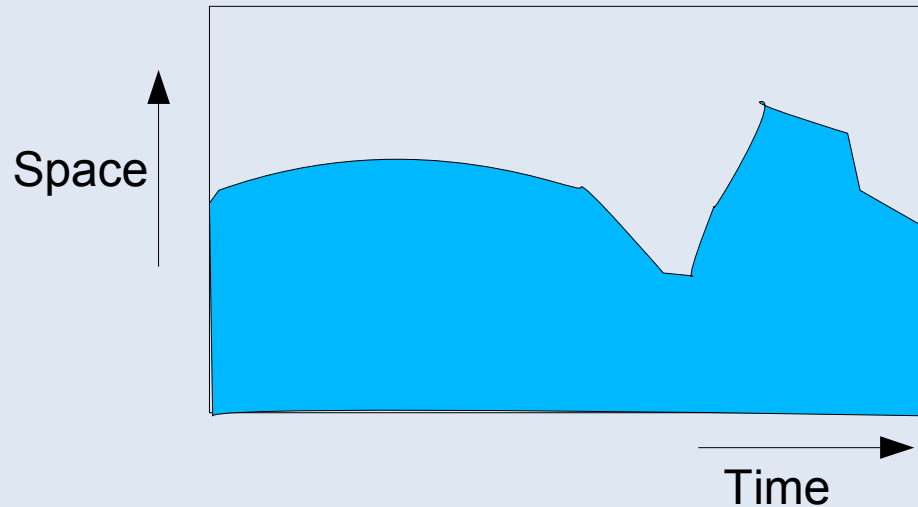
- Why do pages need to be pinned in physical memory?
- Why not pin everything?

Thrashing and Working Sets

- What is thrashing?
 - How do we fix this problem?
- What is a working set?
 - Be able to define this formally
 - Why is it useful?
- What is the Working Set Paging algorithm?
 - Can we do it exactly?
 - How is it implemented?

Space Time Product

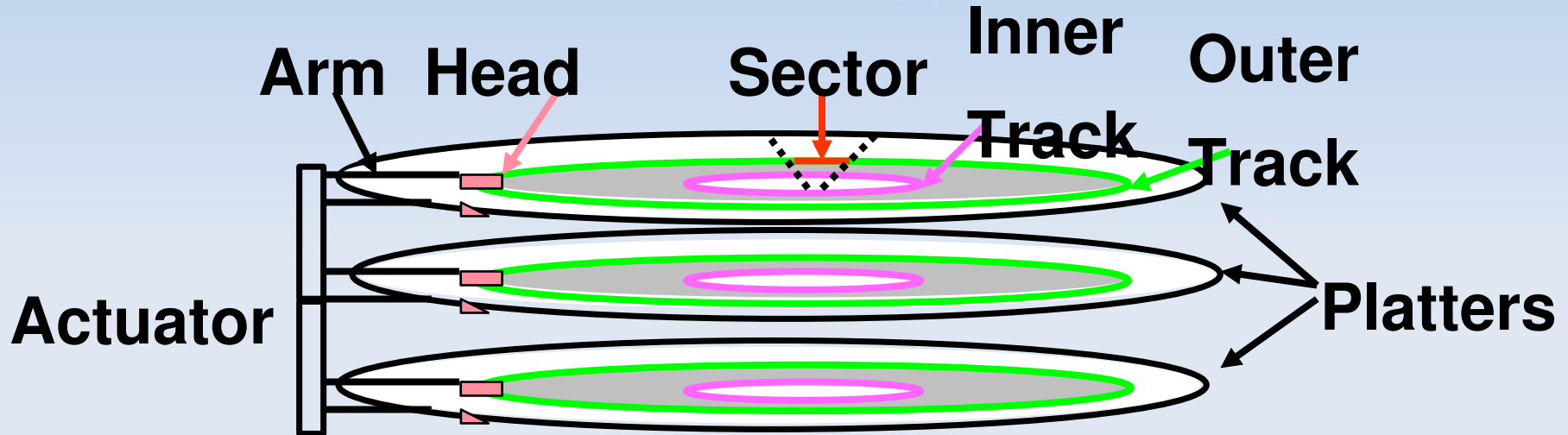
- The amount of memory used by a program needs to be considered over time



I/O Devices

- Know basic characteristics of I/O devices
- Know at least order-of-magnitude
- E.g. desktop hard drives:
 - 10 ms seek time
 - 100 MB/s transfer rate
 - 10,000 RPM
 - 100 GB
 - 10 MB cache

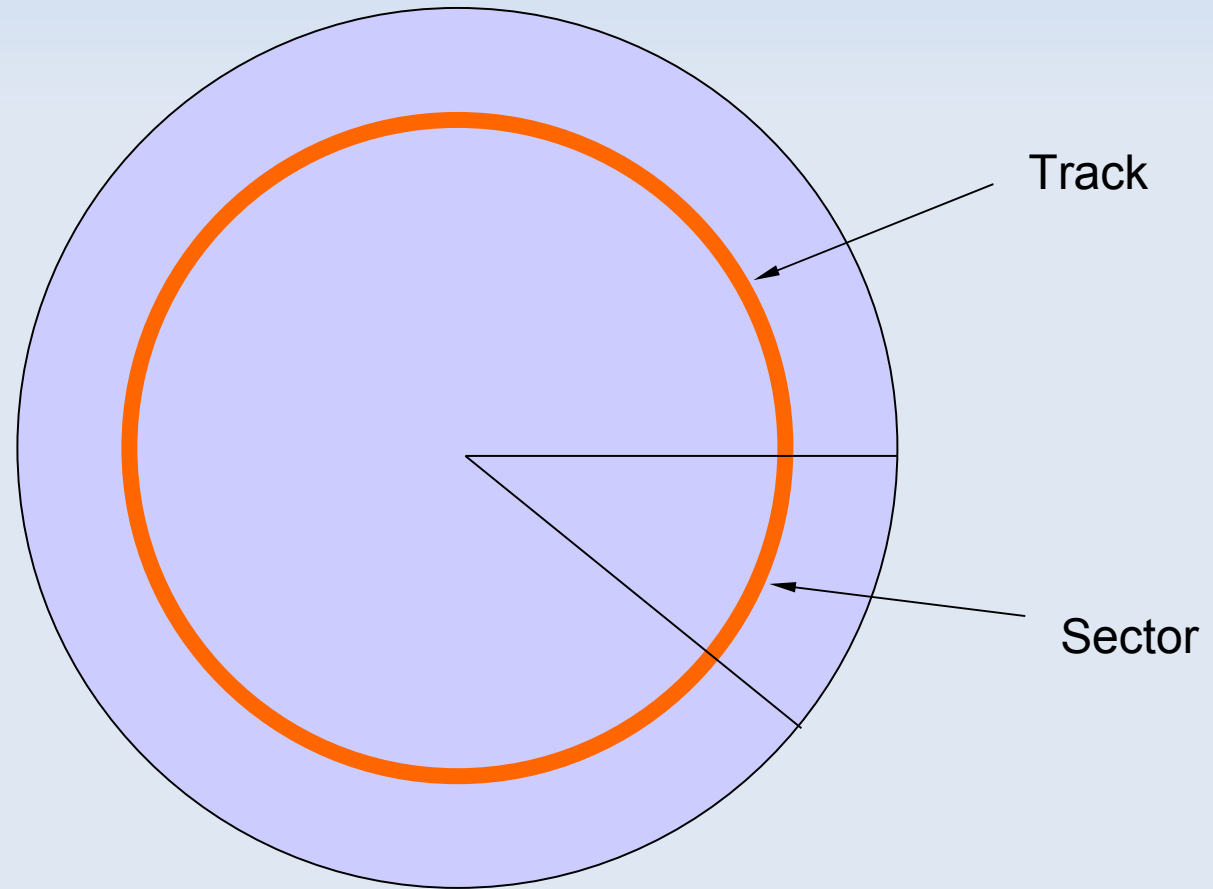
Disk Device Terminology



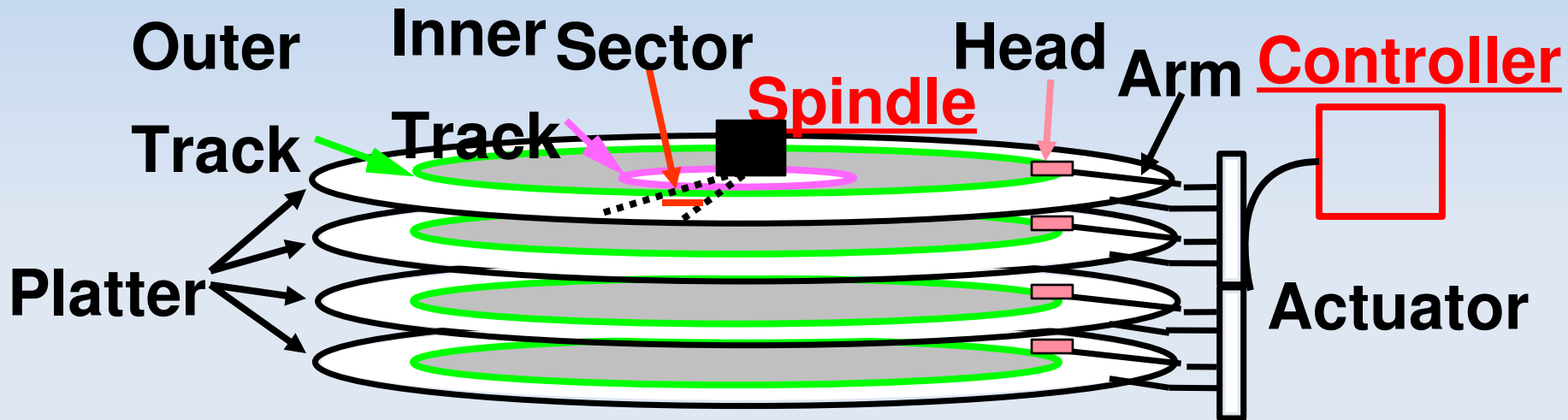
- Several platters, with information recorded magnetically on both surfaces (usually)
- Bits recorded in tracks, which in turn divided into sectors (e.g., 512 Bytes)
- Actuator moves heads (end of arm, 1/surface) over track (“seek”), select surface, wait for sector rotate under head, then read or write

○ “Cylinder”: all tracks under heads

Disk surface



Disk Device Performance



- **Disk Latency = Seek Time + Rotation Time + Transfer Time + Controller Overhead**
- **Seek Time?** depends on number of tracks to move the disk arm and seek speed of disk
- **Rotation Time?** depends on speed disk rotates, how far sector is from head
- **Transfer Time?** depends on data rate (bandwidth) of disk (bit density), size of request

File System and disk management

Physical Reality	Filesystem Abstraction
Block oriented	Byte oriented
Physical sector #	Named files
No protection	Protection
Data maybe corrupted if machine crashes	Robust to machine failure

File system components

- Disk management: how to arrange collection of disk blocks into files
- Naming: user gives file name, not track 50, platter 5, etc.
- Protection: keep information secure
- Reliability/durability: when system crashes, lose stuff in memory, but want files to be durable.

File Usage and Access Patterns

- **Access**

- Sequential access – bytes read in order (give me the next X bytes, then give me next)
- Random access – read/write element out of middle of array
- Keyed access – “find me 100 bytes starting with SMITH”

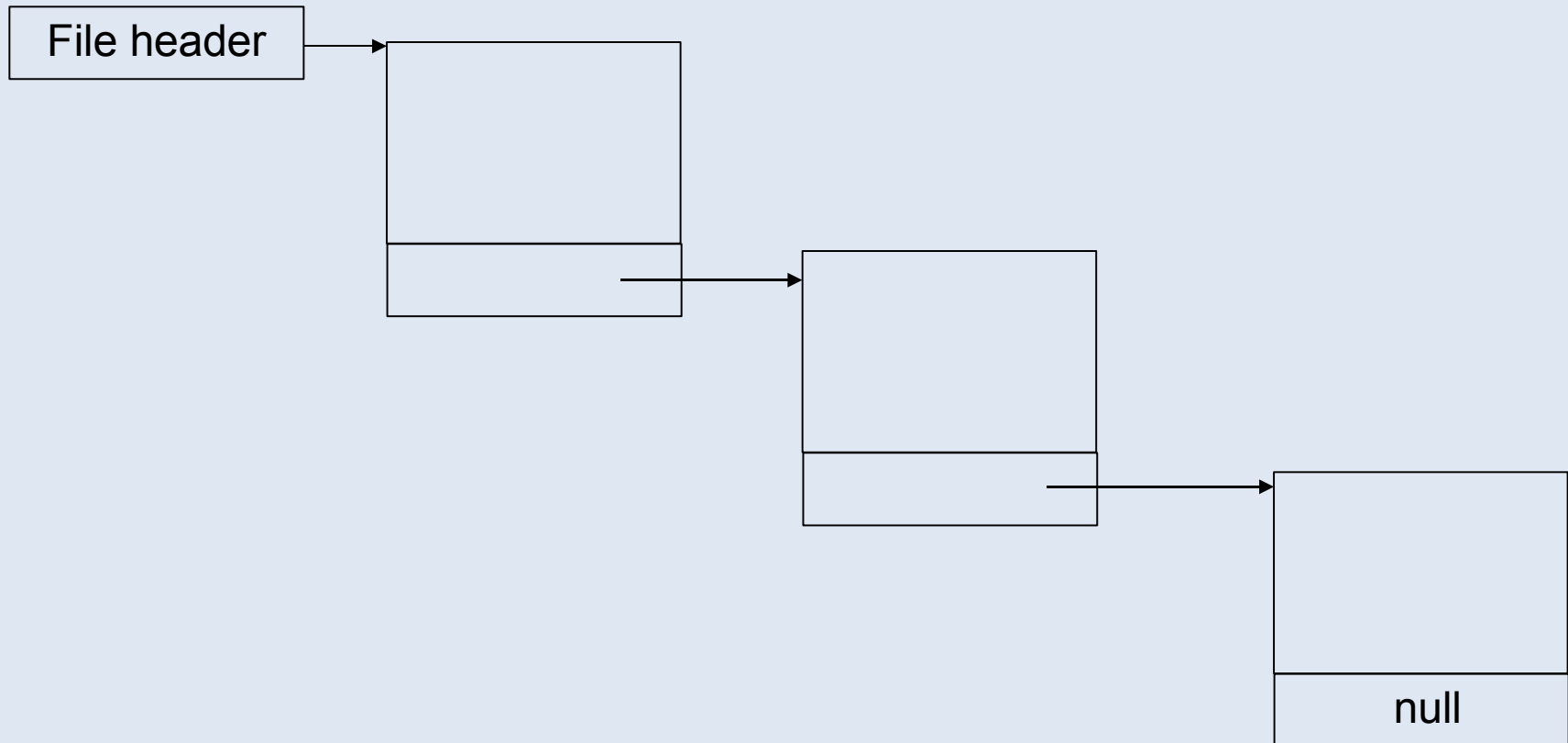
- **Usage**

- Most files are small
- Large files use up most of the disk space
- Large files account for most of the bytes transferred to/from disk
- Most I/O are read and sequential

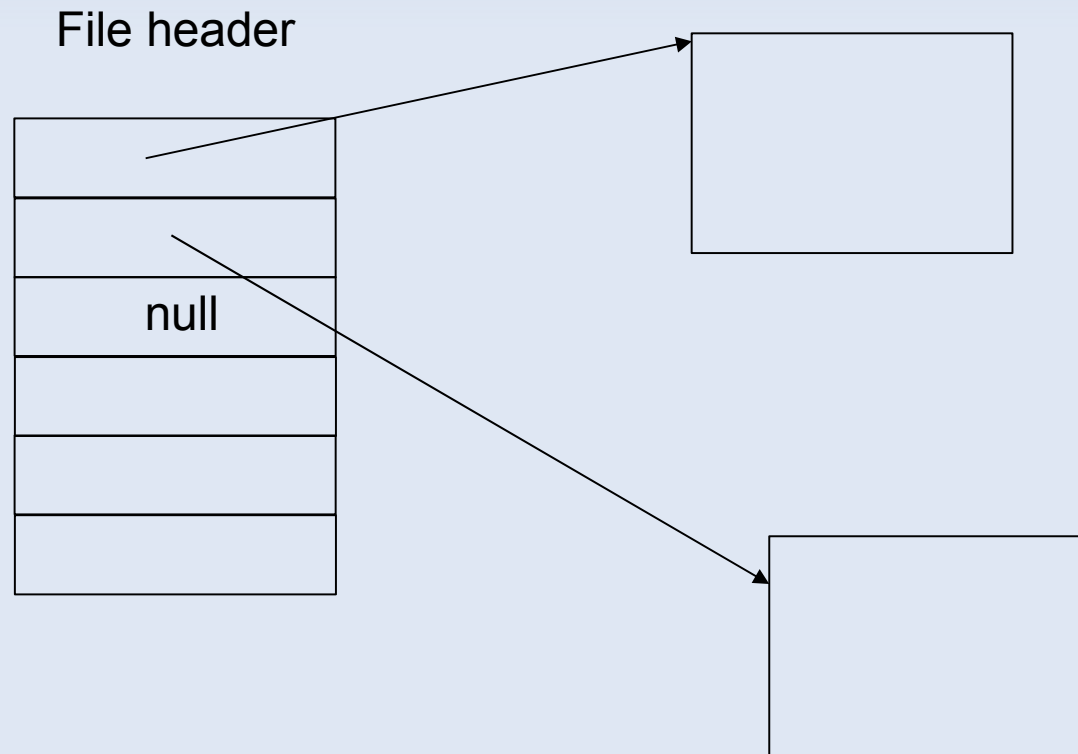
File block layout and access

- Contiguous allocation
- Linked files
- Indexed files
- Multi-level Indexed files
- Should know
 - How do they work?
 - Advantages?
 - Disadvantages?

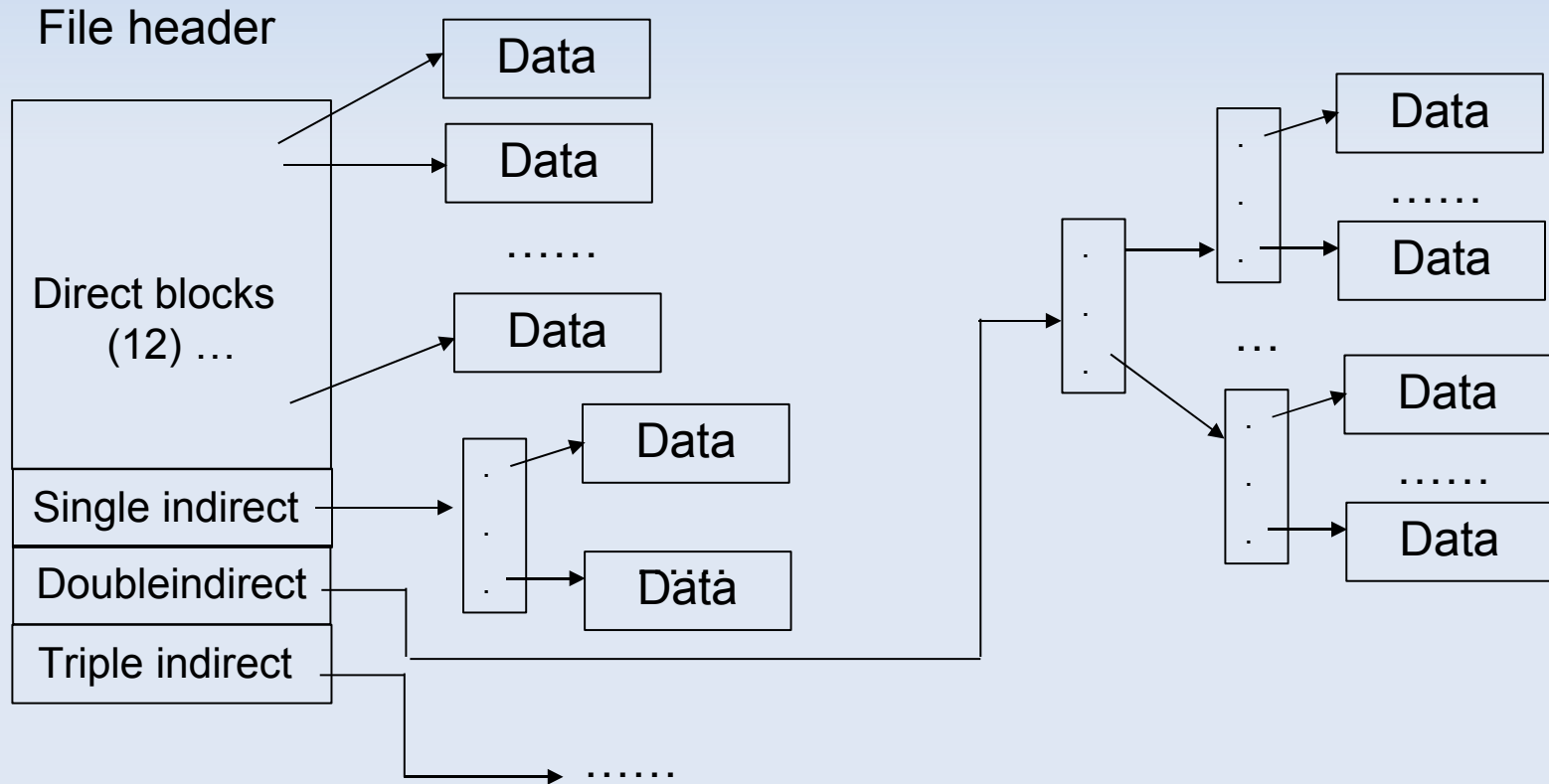
Linked file (Alto)



Indexed file (VMS)



Multi-level Indexed file (4.4BSD)



Given the block size, block pointer size, be able to calculate the max file size.
Be able to show how to access a certain block in a file, how many disk accesses are involved and what they are.

More on file blocks ...

- What is a File Descriptor? (=inode=file header) And how to find them?
 - Stored in a special area on disk in fixed size arrays per cylinder group (unix)
- What information does an inode contain?
 - Num links, owner, group, times, etc.
- Block allocation
 - How do we manage free blocks?
 - How do we allocate a new block for a file?

Naming and directories

- How does a user identify a file?
- How does the OS find its inode?
- Directory
 - Maps file names to inode numbers
 - A table of <file name, file index> pairs
 - Typically organized into a hierarchy
 - / is always at a fixed spot on disk
- Given a directory hierarchy, be able to show how a file is accessed (e.g. calculate how many disk I/Os are needed to read its first byte)

I/O Optimization

- **Block size**
- **Disk arm scheduling**
 - FCFS, SSTF, SCAN, C-SCAN
 - Know the differences, pros & cons of each one of these
- **File placement**
 - Put commonly used files near the center of the disk
 - Group files that are frequently referenced together
 - Put commonly used files on different disks
- **Disk caching**
 - Keep recently used file blocks in main memory
 - Can do read ahead and write behind
 - Also done in the disk controller

I/O Optimization, cont ...

- What can we do at the file system level to increase performance?
 - Pre-fetching?
 - Data Reorganization? – make the physical layout reflect the logical organization
 - Data Replication?
- What are the pros and cons of each of these?

RAID

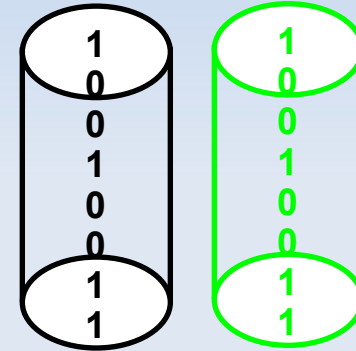
- What is RAID?
- What motivated RAID?
- How does RAID work?
- RAID 0 – striping
- RAID 1 – replication (mirroring)
- RAID 2 – Hamming ECC
- RAID 3/4 – byte/block parity on a fixed disk
- RAID 5 – distributed (interleaved) parity

RAID Techniques Summary

Each disk is fully duplicated onto its "shadow"

Logical write = two physical writes

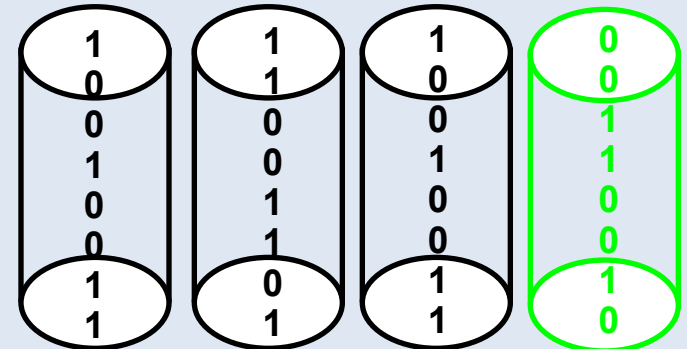
100% capacity overhead



- *Parity Data Bandwidth Array (RAID 3)*

Parity computed horizontally

Logically a single high data bw disk

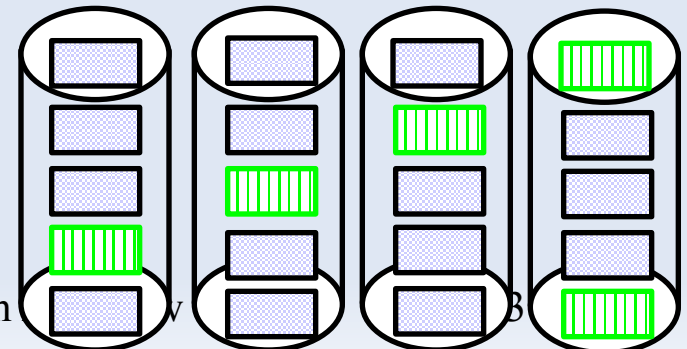


- *High I/O Rate Parity Array (RAID 5)*

Interleaved parity blocks

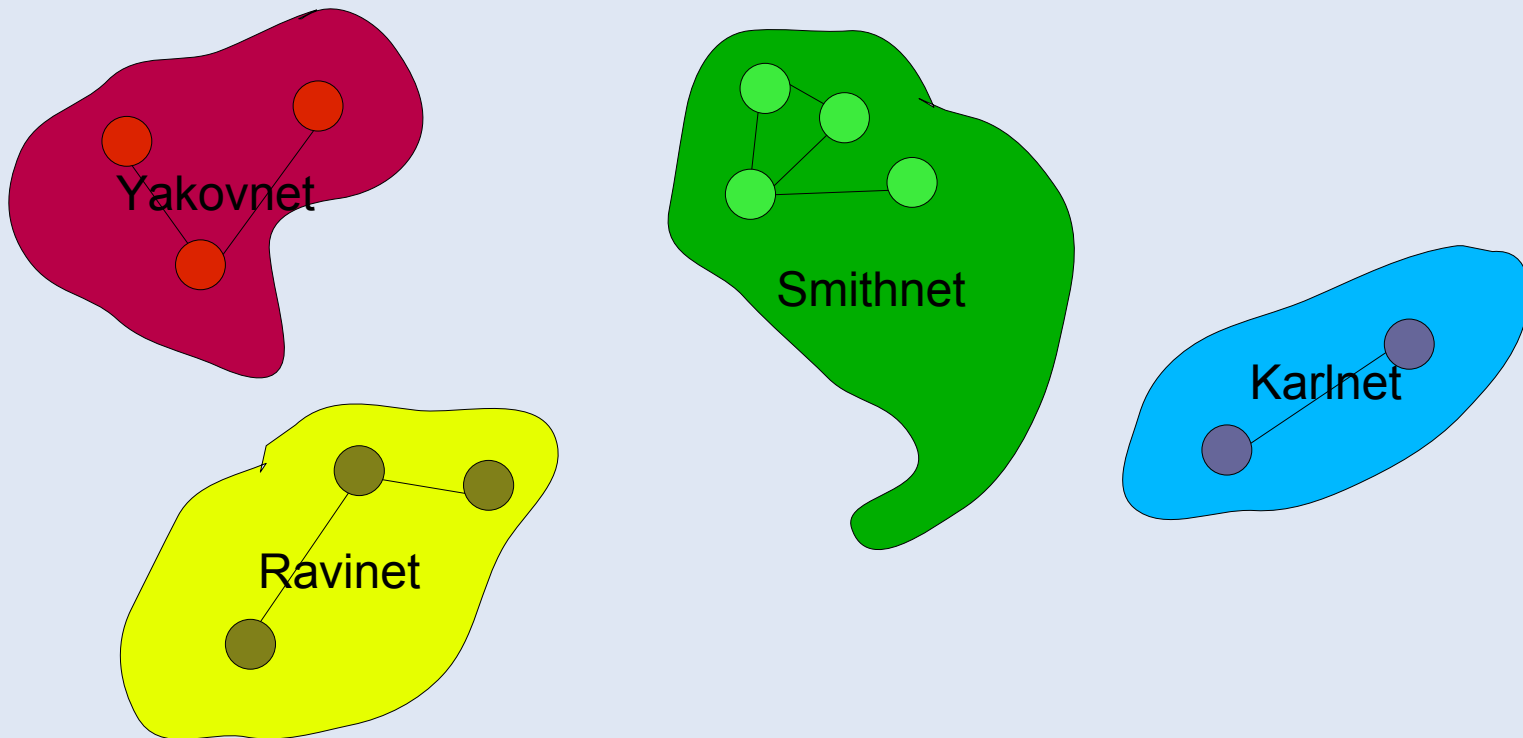
Independent reads and writes

Logical write = 2 reads + 2 writes



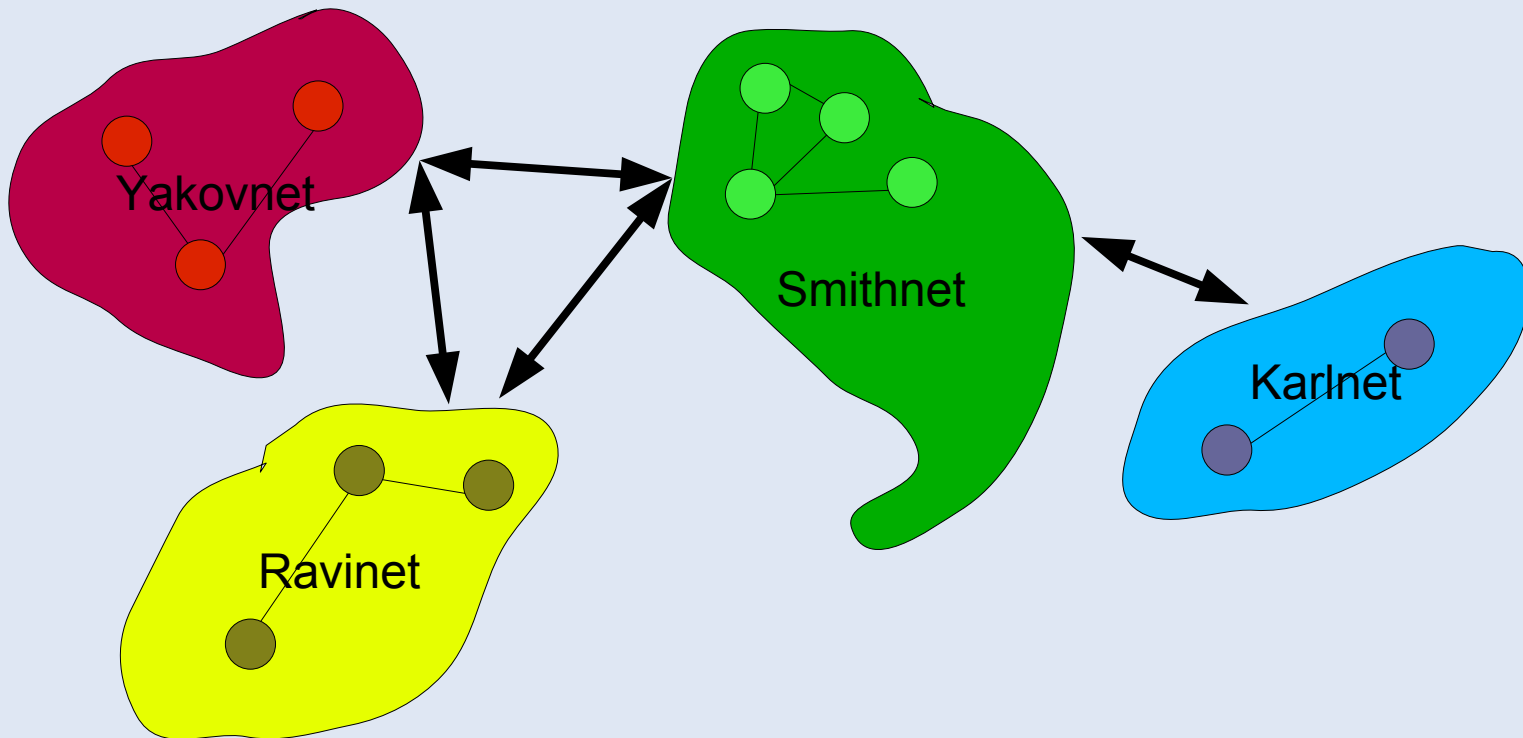
Networking

- Once upon a time: individual LANs



Networking

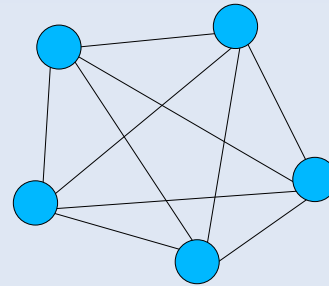
- Why not inter-network them together -> WAN



Networking

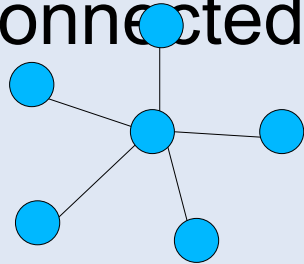
- Topology

- Fully-connected

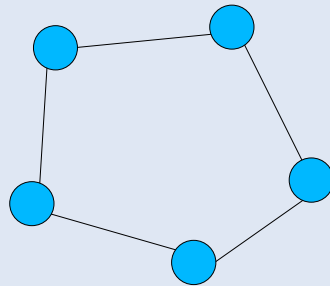


- Partially-connected

- Star



- Ring



Networking

- Broadcasting
 - Radio (Wifi)
 - Ethernet: Hubs (compare switches)
 - Aloha net
 - Can everyone talk at once?
 - Carrier sense
 - Collision detection

Networking

- LANs
 - Ethernet
 - Token-ring
- WANs
 - Circuit switching
 - Packet switching

Networking

- Performance parameters
 - Latency
 - Bandwidth
- Example: DSL
 - Latency = 20 ms
 - Bandwidth = 80 Kbyte/s

Networking

- OSI layers
 - Physical
 - Data link
 - Network
 - Transport
 - Session
 - Presentation
 - Application
- Where do these go:

Networking

- Networks are sparsely connected
- Network layer (e.g. IPv4)
 - Addressing
 - Route packets to destination

Networking

- Network links are unreliable
 - Dropped packets
 - Congestion
- Transport layer (e.g. TCP)
 - Reliability
 - Flow control
 - Congestion control
 - Ports

Sample questions

Disk head scheduling

- Disk just serviced 61 and previously serviced 17
- Requests to the following cylinders outstanding:
 - 200, 340, 2, 98, 112, 55, 499
- What is the order of requests for
 - FCFS, SSTF, SCAN, C-SCAN?

Page replacement

- Given this sequence of page requests:
 - 5 4 3 2 5 4 6 5 4 3 2 6
- What is the number of page faults for:
 - MIN, FIFO, LRU, Clock

Disk drives and file systems

- Consider a fixed-head disk drive. Which of these are no longer important?
 - Cylinder groups
 - Bitmap free list

File system

- Suppose I want to read the 12,523th byte of
 - `/home/cs162-aa/nachos/threads.java`
- List which disk blocks must be read into memory
 - Block size: 1024 bytes
 - Inode: 10 direct pointers, two 2-level pointers, two 3-level pointers
 - Assume directories all fit onto one disk block