SE 292: High Performance Computing [3:0][Aug:2014]

File Systems

Yogesh Simmhan

Adapted from:

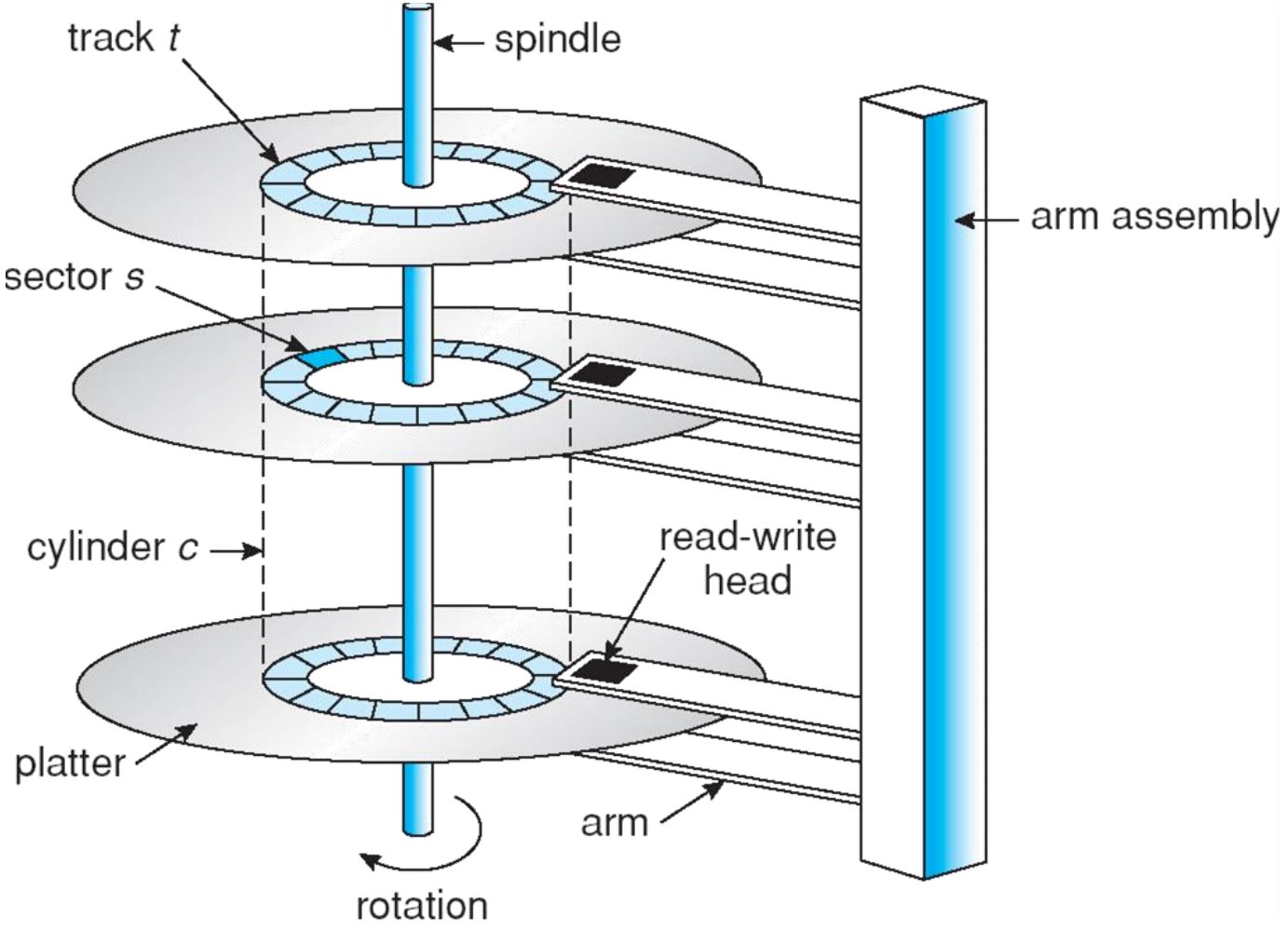
- *"File Systems", Sathish Vadhiyar, SE292 (Aug:2013),*
- *"Storage: Where it's come from and is going", Christos Papadopoulos*

File Systems

What is a file?

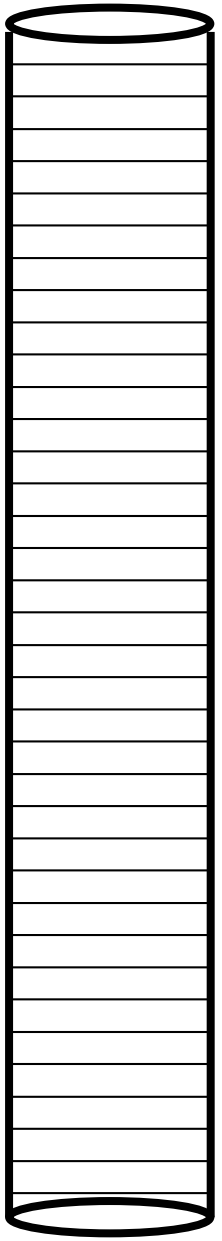
- Storage that continues to exist beyond lifetime of program (persistent)
- Named sequence of bytes stored on disk

Moving-head Disk Mechanism (HDD)

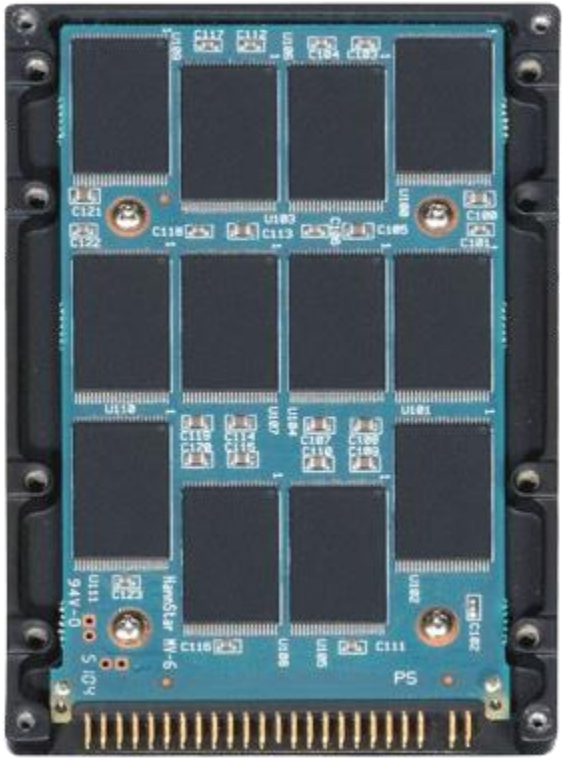


About HDD

- Platter: metal disk covered with magnetic material
- Multiple platters rotating together on common spindle
- Read/write head: electromagnet used to read/write
- Tracks: concentric circular recording surfaces
- Sector/block: unit of track that is read/written
- Head associated with disk arm, attached to actuator
- Cylinder: all tracks associated with a given actuator position
- Our view of disk: linear address space of fixed size sectors/blocks numbered from 0 up



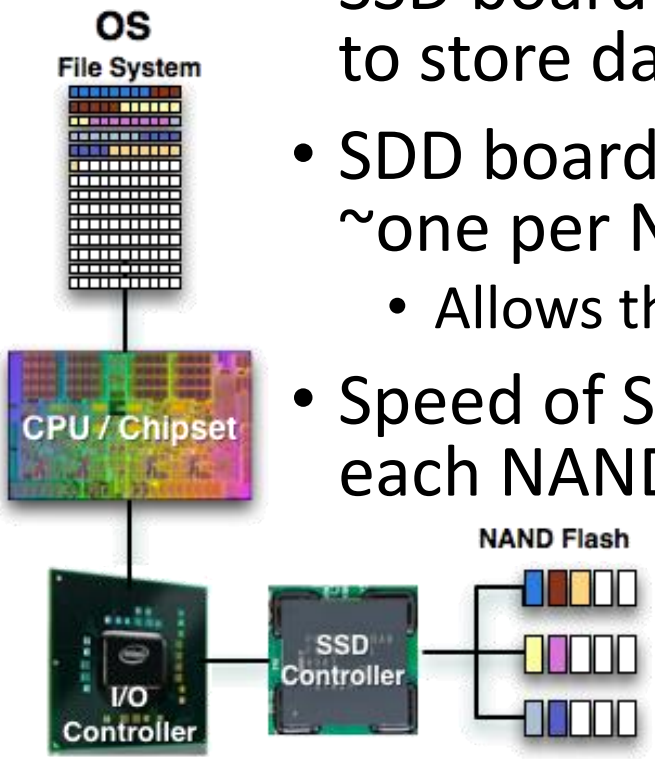
HDD vs SSD



appleinsider.com

Solid State Drives (SSD)

- Technology is used in USB Flash Drives
- Use integrated chips (IC) for storage
  - Why?
- SSD board contains number of NAND ICs used to store data
- SDD board also contains support channels, ~one per NAND IC
  - Allows the controller to communicate with each IC
- Speed of SSD comes from parallel access to each NAND IC



SSD vs HDD Performance

Drive	Read (MB/s)	Write (MB/s)	Size (GB)	Cost per drive	Cost per GB	Cost per MB/s
512GB SSD	584	551	512	$469	$0.92	$0.41
256GB SSD	650	551	256	$230	$0.90	$0.19
128GB SSD	646	362	128	$130	$1.02	$0.13
240GB SSD	533	549	240	$240	$1.00	$0.22
120GB SSD	531	538	120	$129	$1.08	$0.12
60GB SSD	519	523	60	$84.50	$1.41	$0.08
1TB SATA	158	137	1000	$86	$0.09	$0.29

http://www.advancedclustering.com/hpc-cluster-blog-ssd-vs-hdd/

SSD

- Pros
  - Almost instantaneous read and write times
  - The ability to read or write in multiple locations at once
  - The speed of the drive scales extremely well with the number of NAND ICs on board
  - No moving parts
- Cons
  - To erase the value in flash memory the original voltage must be reset to neutral
  - We have to delete an entire block to release it
  - Can only be erased 10,000 times before it goes bad
  - Writes can be slower, SSD lifetime can be low.
  - 10x costlier than HDD

Other Disk Components

- Disk drive is connected to computer by I/O bus
- Data transfers on bus carried by special processors – host controller on the host side, disk controller on the disk side

Disk Performance

- Transfer rate
  - Rate of data flow between disk drive and computer (few megabytes per sec)
  - Data transferred from memory to disks in units of blocks. Each block consists of sectors.
  - EIDE/(P)ATA: 3-167MB/s
  - SATA: 150-600MB/s

Disk Performance

- Seek time/latency – time to move HDD disk arm to desired cylinder (few milliseconds)
- Rotational time/latency – time for the sector in the track to rotate and position and under the head (few milliseconds)

Disk Attachment

- Can be host-attached – DVD, CD, hard disk by special buses and protocols
  - Protocols - SATA, SCSI (difference in terms of number of disk drives, address space, speed of transfers)
- Network-Attached – NFS
- Storage Area Network
  - To prevent storage traffic interfering with other network traffic
  - Specialized network
  - Has flexibility regarding connecting storage arrays and hosts

Operations on Files

- fd = open (name, operation)
- fd = create (name, mode)
- status = close(fd)
- bytecount = read (fd, buffer, bytecount)
- bytecount = write (fd, buffer, bytecount)
- offset = lseek (fd, offset, whence)
- status = link (oldname, newname)
- status = unlink (name)
- status = stat (name, buffer)
- status = chown (name, owner, group)
- status = chmod (name, mode)

Common File Access Patterns

- Sequential access: bytes of file are read in order from start to finish
- Random access: bytes of file are read in some (random) order

File System Design Issues

- Disk management: efficient use of disk space
- Name management: how users select files for use
- Protection: of files from users

Disk Management

Issues

1. Allocation: How are disk blocks associated with a file?
2. Arm scheduling: Which disk I/O request should be sent to disk next?

   FCFS, Shortest Seek Time First (SSTF), Scan, C-Scan

File Descriptor: OS structure that describes which blocks on disk represent a file

Disk Block Allocation: Contiguous

File is stored in contiguous blocks on disk

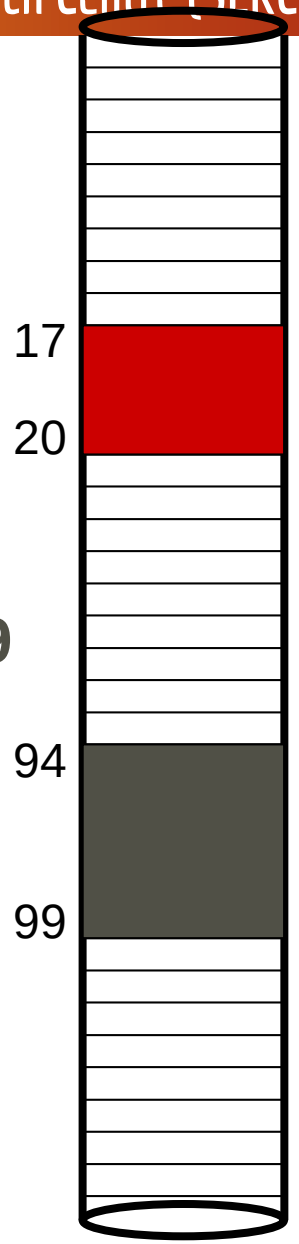
- File descriptor: first block address, file size

File 1: Size 4 blocks; Blocks 17, 18, 19, 20

File 2: Size 6 blocks; Blocks 94, 95, 96, 97, 98, 99

File 1: Start 17 Size 4

File 2: Start 94 Size 6



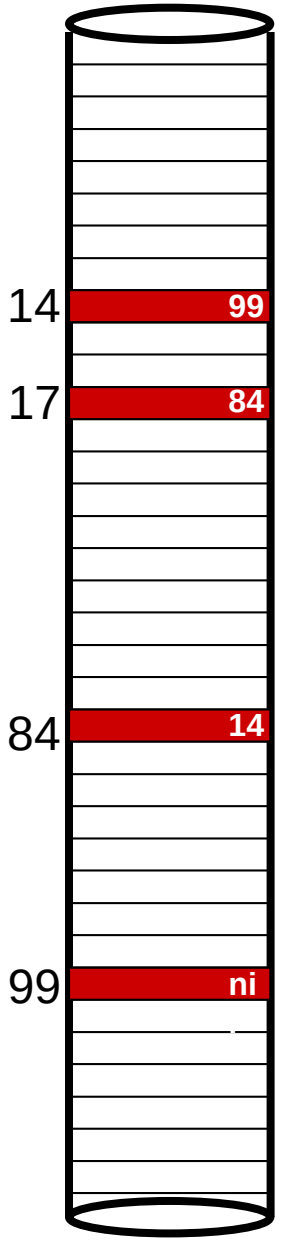
Disk Block Allocation: Linked

Each block contains disk address of next file block

- File descriptor: first block address

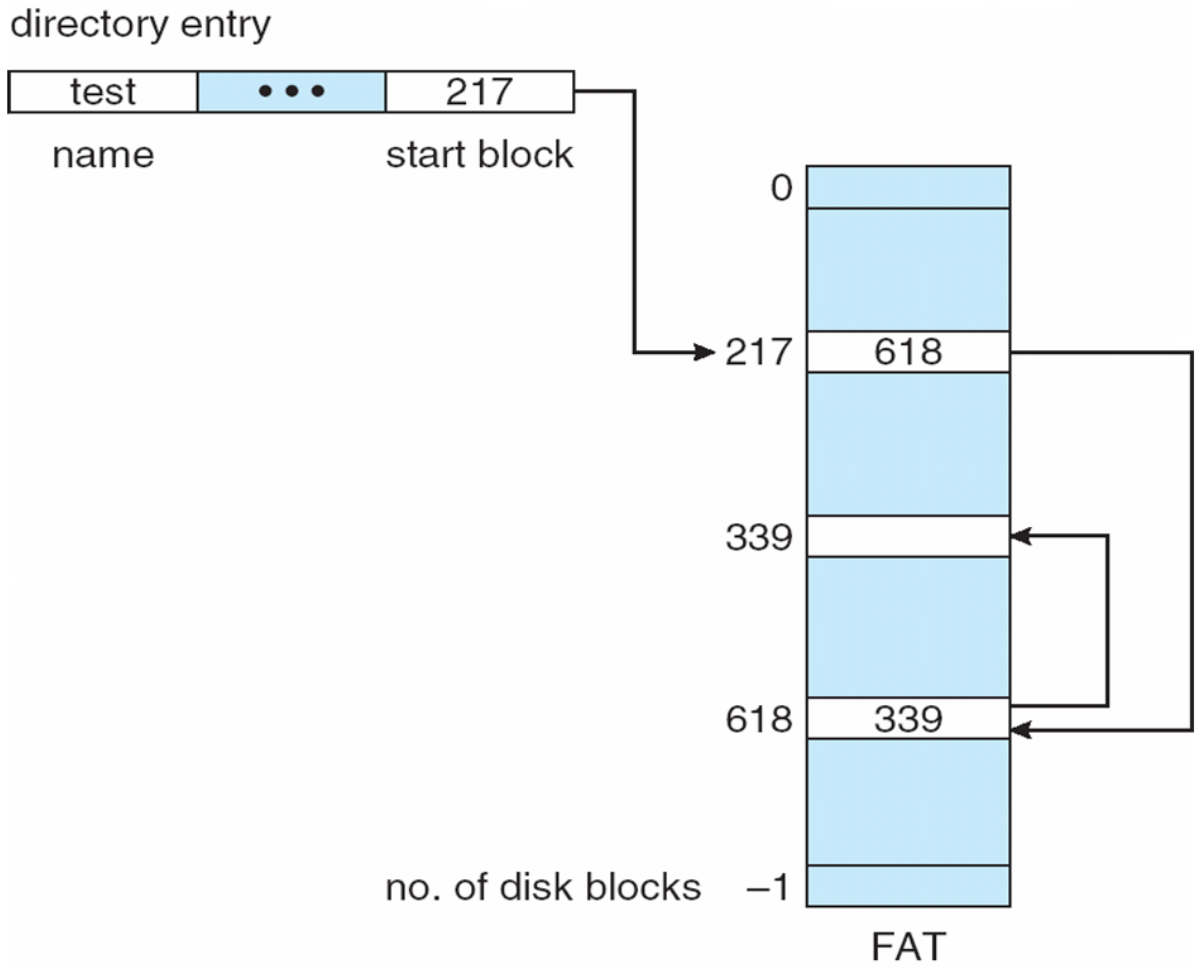
File 1: Size 4 blocks; Blocks 17, 84, 14, 99

File 1: Start 17



FAT system

- File Allocation Table
- A form of indexed allocation
- A portion of disk used for FAT



Disk Block Allocation: Indexed

File Index is an array containing addresses of 1st, 2nd, etc block of file

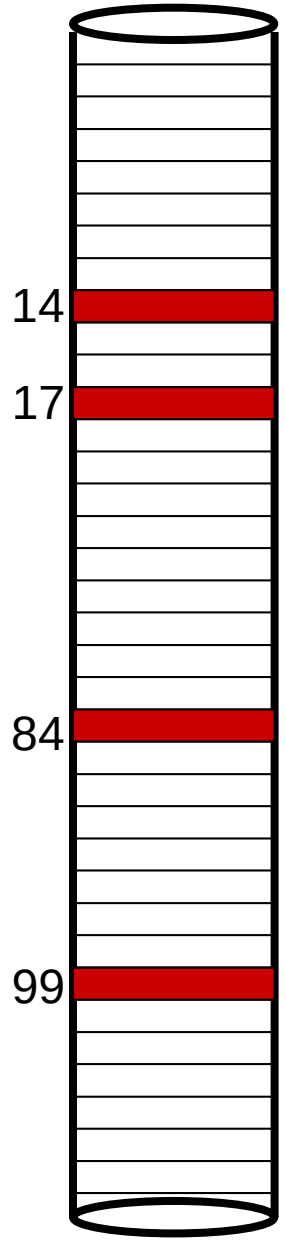
- File descriptor: index

File 1: Size 4 blocks; Blocks 17, 84, 14, 99

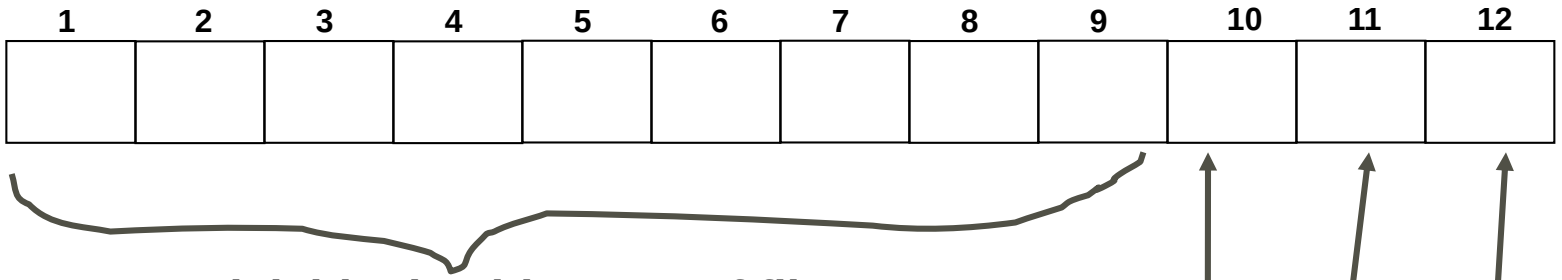
	1	2	3	4
INDEX	17	84	14	99

Problem: size of the index?

Some schemes?



UNIX Version of Indexed Allocation



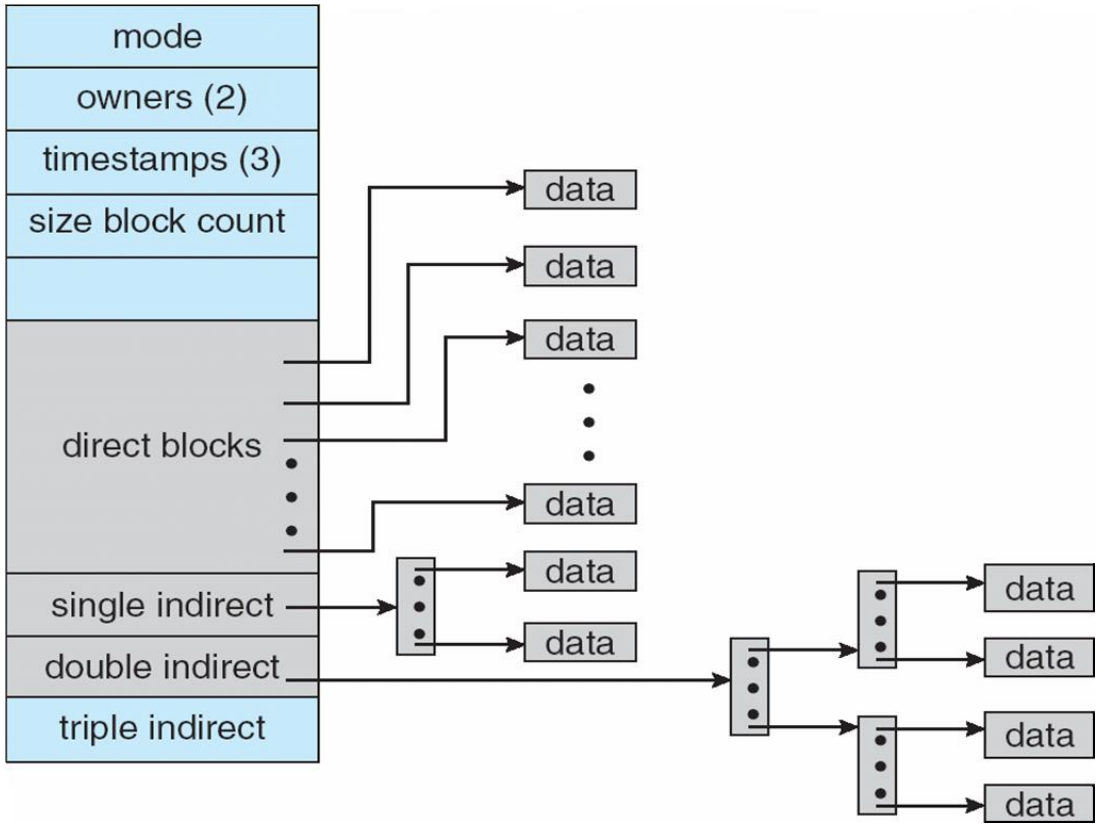
Assume disk block size: 1KB,

disk block address size: 4B

Maximum file size:

9KB + 256KB
+ 256*256 KB
+256*256*256 KB

Combined Scheme: UNIX (4K bytes per block)



- Pointers can occupy significant space
- Performance can be improved – disk controller cache, buffer cache

Name Management

Issues: How do users refer to files? How does OS find file, given a name?

- Directory: mapping between file name and file descriptor
- Could have a single directory for the whole disk, or a separate directory for each user
- UNIX: tree structured directory hierarchy
  - Directories stored on disk like regular files
  - Each contains (filename, i-number) pairs
  - Each contains an entry with name **.** for itself (**..**)
  - Special (nameless) directory called the root

MTech Projects@DREAM:Lab

- Big Data Platforms and Infrastructure
  - **Graph programming models, analytics, algorithms**
  - *Apache Giraph/Google Pregel/GoFFish*
  - **Stream and Complex event processing platforms**
  - *Apache/Twitter Storm, Internet of Things Apps*
- Cloud Computing
  - **Dynamic & Adaptive Job Scheduling on Clouds**
  - Amazon EC2, OpenStack Cloud
- Commodity Private Cloud
  - 224 Core AMD Opteron Cluster, 24 nodes*8 cores
  - SSD, HDD, GbE
  - Hadoop, Giraph, Storm, OpenStack Cloud, PBS, …

Protection

Objective: to prevent accidental or intentional misuse of a file system

- Aspects of a protection mechanism
  - User identification (authentication)
  - Authorization determination: determining what the user is entitled to do
  - Access enforcement
- UNIX
  - 3 sets of 3 access permission bits in each descriptor

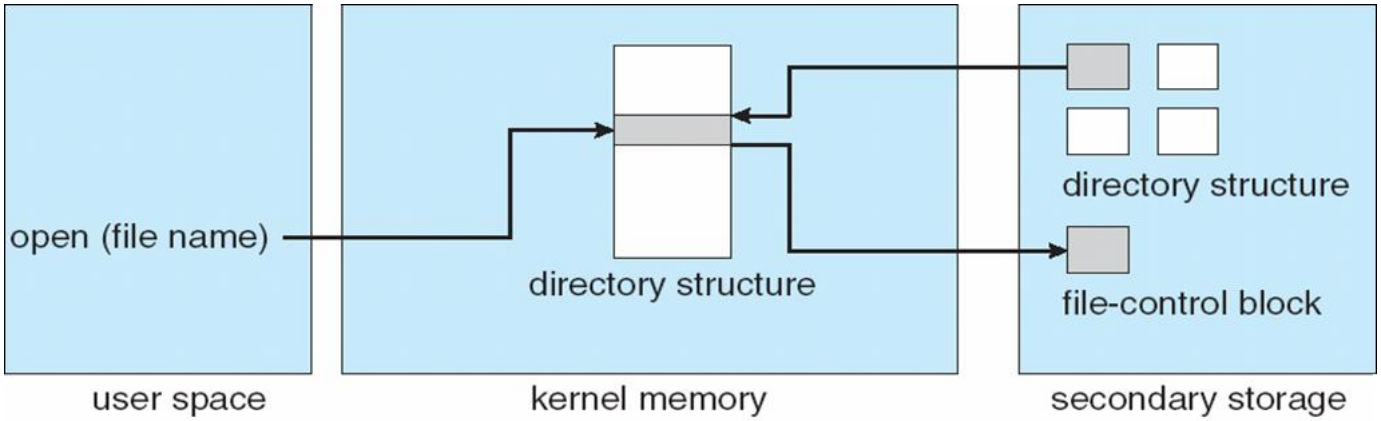
File System Structure

- Layered file structure consisting of following layers (top to bottom)
- Logical file system
  - contains inodes or file control block – a FCB contains information about file including ownership, permission, location
- File organization
  - Translation between logical and physical blocks
- Basic file system
  - manages buffers and caches
- I/O control
  - contains device driver
- Devices

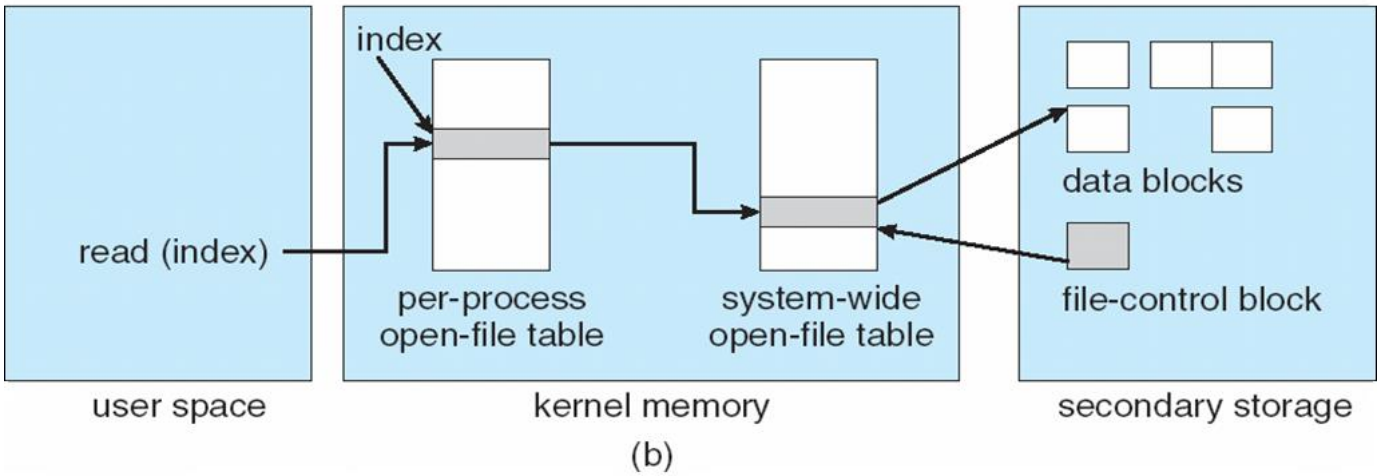
File System Implementation

- In disks – FCB (contains pointers to blocks)
- In memory – system-wide open file table, per-process file table (thus 2 tables)
- Operations on file using pointer to an entry in per-process file table
- Entry is referred as file descriptor

In-Memory File System Structures

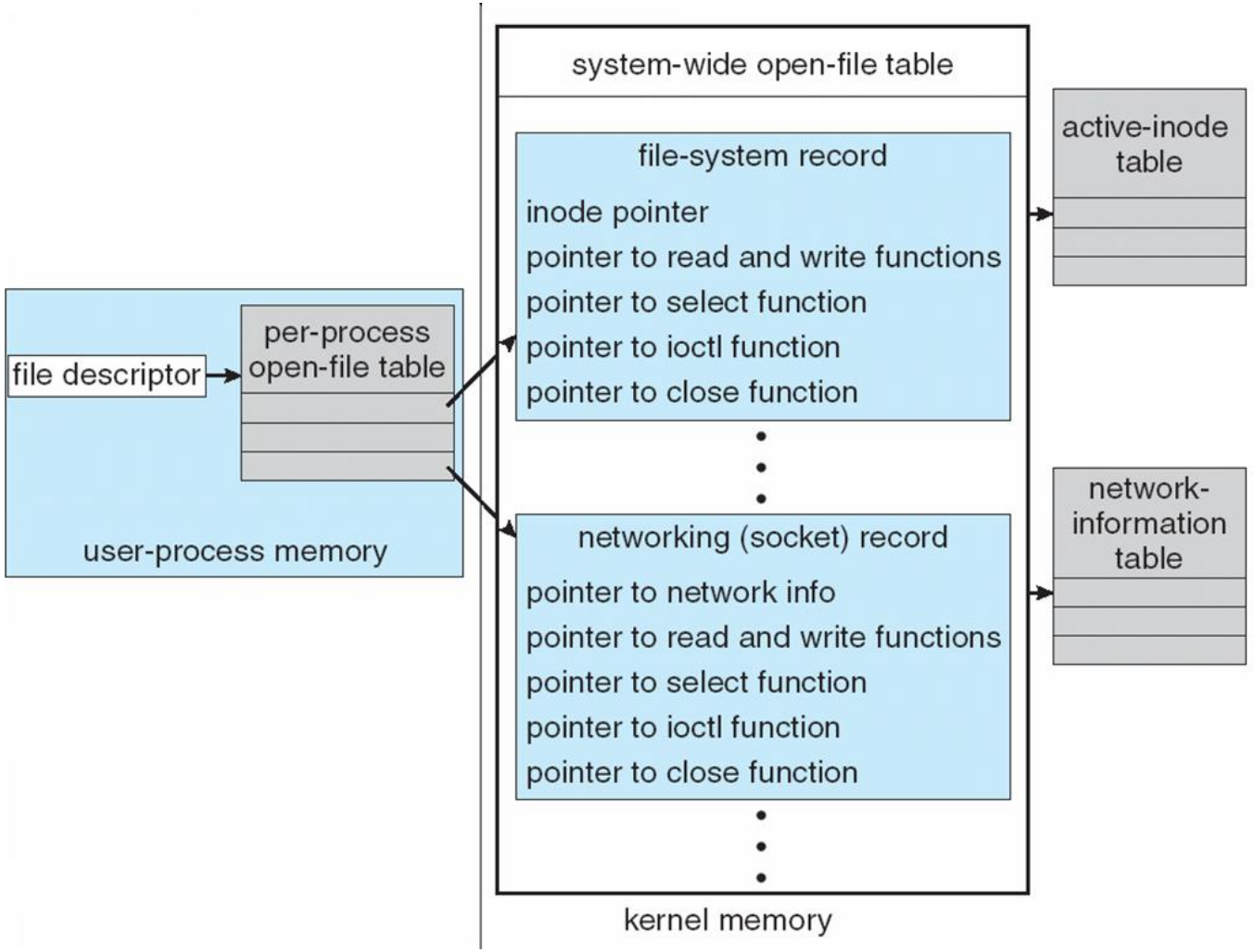


File Open

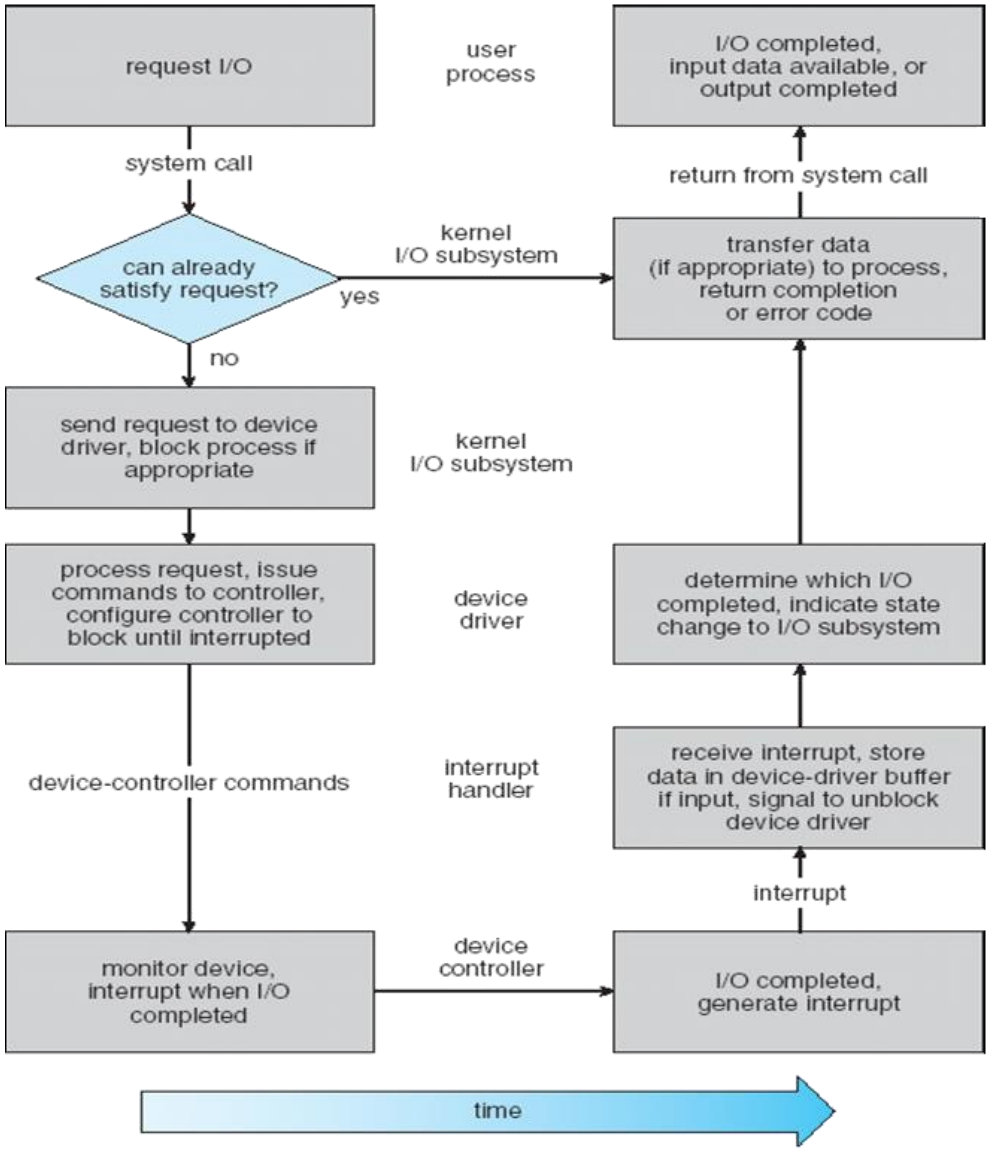


File Read

UNIX I/O Kernel Structure



Life Cycle of An I/O Request



File System Performance Ideas

- Caching or buffering
  - System keeps in main memory a disk cache of recently used disk blocks
  - Could be managed using an LRU like policy
- Pre-fetching
  - If a file is being read sequentially, a few blocks can be read ahead from the disk

Memory Mapped Files

- Traditional open, /lseek/read/write/close are inefficient due to system calls, data copying
- Alternative: map file into process virtual address space
- Access file contents using memory addresses
- Can result in page fault if that part not in memory
- Applications can access and update in the file directly and in-place (instead of seeks)
- System call: `mmap(addr,len,prot,flags,fd,off)`
- Some OS's: cat, cp use mmap for file access

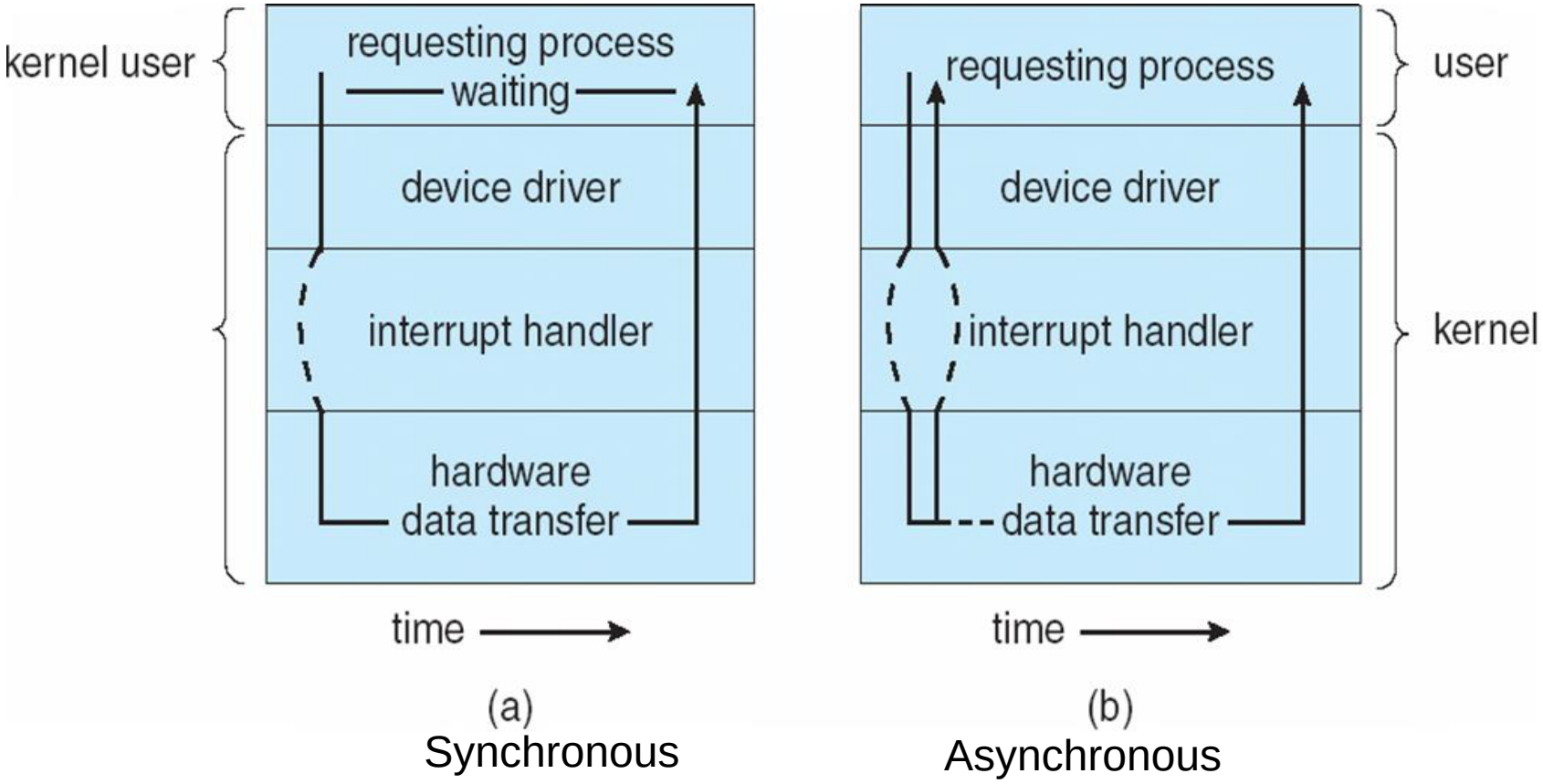
Asynchronous I/O

- Objective: allows programmer to write program so that process can perform I/O without blocking
- Eg: SunOS aioread, aiowrite library calls
  - Aioread(fd, buff, numbytes, offset, whence, result)
    - Reads numbytes bytes of data from fd into buff from position specified and offset
    - The buffer should not be referenced until after operation is completed; until then it is in use by the OS
    - Notification of completion may be obtained through aiowait or asynchronously by handling signal SIGIO

Blocking and Non-Blocking I/O

- Blocking – process moved from ready to wait queue. Execution of application is suspended.
- Non-blocking – overlapping computation and I/O. Using threads.

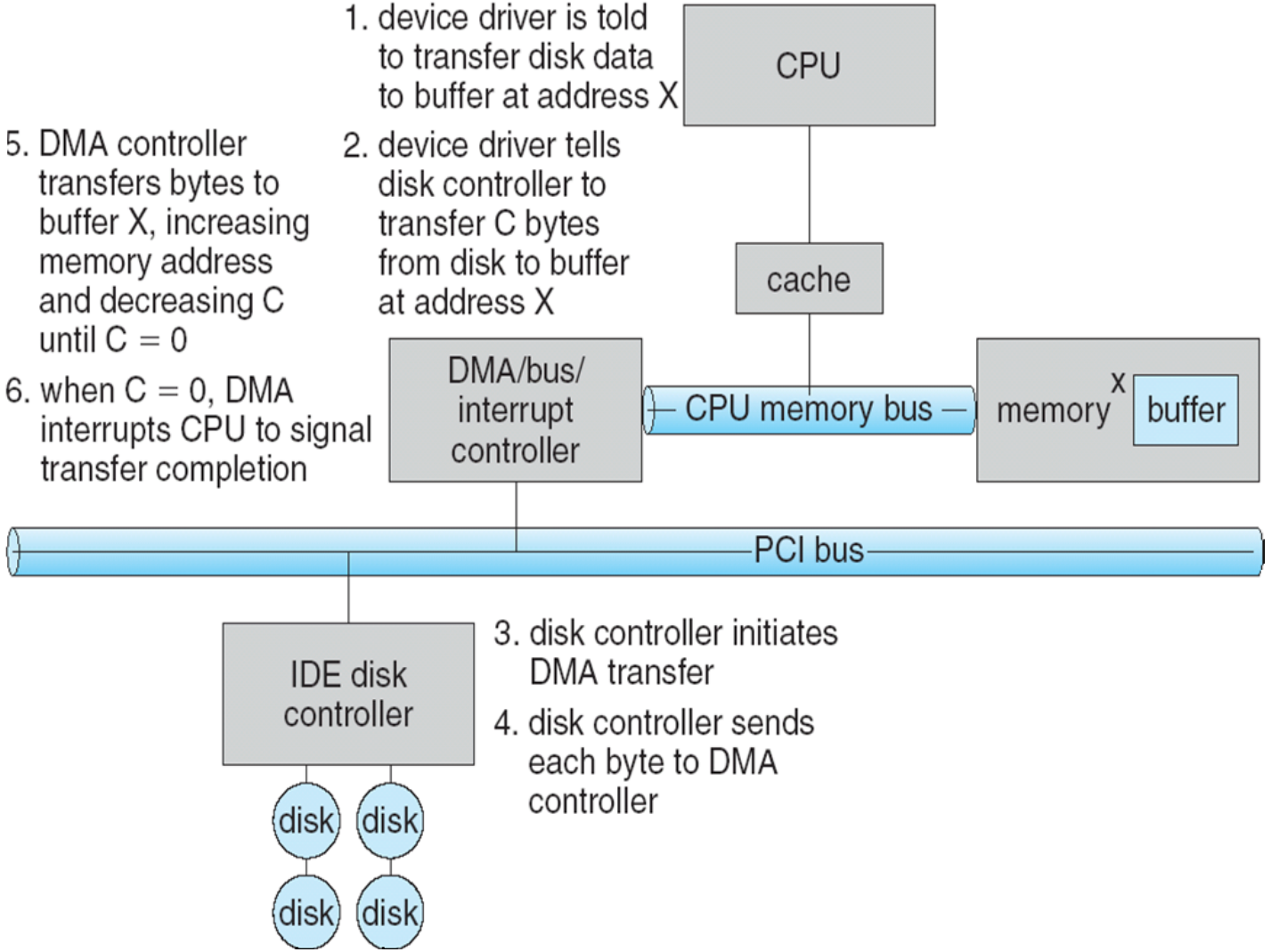
Two I/O Methods



DMA (Direct Memory Access)

- It is wasteful for the CPU to engage in I/O between device and memory
- Many systems have special purpose processor called DMA controller
- CPU writes "I/O details" to memory
- Sends this address to DMA controller
- Thereafter, DMA engages in transfer of data between device and memory
- Once complete, DMA controller informs (interrupts) CPU

Six Step Process to Perform DMA Transfer



RAID

- Redundant Array of Independent Disks (RAIDS) – multiple disks to improve performance and reliability
- Reliability
  - MTBF (Mean Time Between Failure) decreases with more disks
  - Hence data has to be redundantly stored
- Performance
  - Can be used to increase simultaneous access and transfer rate (striping)

Reading

- File System interface, implementation & mass storage, Silberschatz 7th Ed.
  - Chapter 10,
  - Chapter 11.1-11.4, 11.6
  - Chapter 12.1-12.4,12.7

SE252: Intro to Cloud Computing

1. Cloud computing as a technology
   - How to use Cloud tools, APIs, SDKs?
   - *Access to Amazon AWS IaaS Cloud*
2. Cloud computing as a distributed systems environment
   - Why cloud computing works?
   - How to design applications for Clouds?
3. Cloud computing as a research topic
   - What are the gaps and emerging ideas?

SE252: Intro to Cloud Computing

- 3:1 Course…Programming intensive, Java strongly suggested
- Learning Outcomes
  - Parallel and Distributed Systems Context
  - Cloud Virtualization, Abstractions and Enabling Technologies
  - Algorithms and Big Data Programming for Cloud Applications
  - Application Execution Models on Clouds:
  - Performance, scalability and consistency on Clouds
- Project, Research Writing