TCSS 422: OPERATING SYSTEMS

Beyond Physical Memory, I/O Devices

Wes J. Lloyd
School of Engineering and Technology,
University of Washington - Tacoma

December 3, 2018
TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington   Tacoma

---

FEEDBACK FROM 11/28

- Assignment #3

- A good starting point is to first iterate the set of processes in Linux, and print out the proc ID and name.

- This link, Chapter #3, "The Process Family Tree", should be helpful:

- https://notes.shichao.io/lkd/ch3/

December 3, 2018
TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma
L18.2

FEEDBACK - 2

December 3, 2018
TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma
L18.3

OBJECTIVES

- Quiz 5
- Program 3
- Practice Final – 12/5
- **Paging**
- Chapter 21/22 – Beyond Physical Memory
- Chapter 36 – I/O Devices
- Chapter 37 – Hard Disk Drives

December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.4

CHAPTER 21/22: BEYOND PHYSICAL MEMORY

December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.5
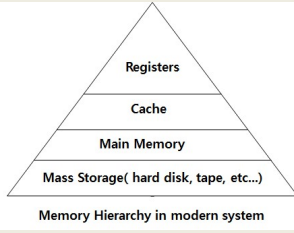

MEMORY HIERARCHY

- Disks (HDD, SSD) provide another level of storage in the memory hierarchy

Registers
Cache
Main Memory
Mass Storage(hard disk, tape, etc...)
Memory Hierarchy in modern system

December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.6

MOTIVATION FOR EXPANDING THE ADDRESS SPACE

- Can provide illusion of an address space larger than physical RAM

- For a single process
  - Convenience
  - Ease of use

- For multiple processes
  - Large virtual memory space for many concurrent processes

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.7 |

LATENCY TIMES

- Design considerations
  - SSDs 4x the time of DRAM
  - HDDs 80x the time of DRAM

Action	Latency (ns)	(µs)	
L1 cache reference	0.5ns		
L2 cache reference	7 ns		14x L1 cache
Mutex lock/unlock	25 ns		
Main memory reference	100 ns		20x L2 cache, 200x L1
Read 4K randomly from SSD*	150,000 ns	150 µs	~1GB/sec SSD
Read 1 MB sequentially from memory	250,000 ns	250 µs	
Read 1 MB sequentially from SSD*	1,000,000 ns	1,000 µs	1 ms ~1GB/sec SSD, 4X memory
Read 1 MB sequentially from disk	20,000,000 ns	20,000 µs	20 ms 80x memory, 20X SSD

- Latency numbers every programmer should know
- From: https://gist.github.com/jboner/2841832#file-latency-txt

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.8 |

SWAP SPACE

- Disk space for storing memory pages
- "Swap" them in and out of memory to disk as needed

Physical Memory and Swap Space

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.9 |

PAGE LOCATION

- Page table pages are:
  - Stored in memory
  - Swapped to disk

- Present bit
  - In the page table entry (PTE) indicates if page is present

- Page fault
  - Memory page is accessed, but has been swapped to disk

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.10

PAGE FAULT

- OS steps in to handle the page fault

- Loading page from disk requires a free memory page

- Page-Fault Algorithm

```
1:      PFN = FindFreePhysicalPage()
2:      if (PFN == -1)                  // no free page found
3:              PFN = EvictPage()       // run replacement algorithm
4:      DiskRead(PTE.DiskAddr, pfn)      // sleep (waiting for I/O)
5:      PTE.present = True              // set PTE bit to present
6:      PTE.PFN = PFN                   // reference new loaded page
7:      RetryInstruction()             // retry instruction
```

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.11

PAGE REPLACEMENTS

- Page daemon
  - Background threads which monitors swapped pages

- Low watermark (LW)
  - Threshold for when to swap pages to disk
  - Daemon checks: free pages < LW
  - Begin swapping to disk until reaching the highwater mark

- High watermark (HW)
  - Target threshold of free memory pages
  - Daemon free until: free pages >= HW

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.12

REPLACEMENT POLICIES

CACHE MANAGEMENT

- Replacement policies apply to "any" cache
- Goal is to minimize the number of misses
- Average memory access time can be estimated:

$$AMAT = (P_{Hit} * T_M) + (P_{Miss} * T_D)$$

Argument	Meaning
T_M	The cost of accessing memory (time)
T_D	The cost of accessing disk (time)
P_{Hit}	The probability of finding the data item in the cache(a hit)
P_{Miss}	The probability of not finding the data in the cache(a miss)

- Consider T_M = 100 ns, T_D = 10ms
- Consider P_{hit} = .9 (90%), P_{miss} = .1
- Consider P_{hit} = .999 (99.9%), P_{miss} = .001

OPTIMAL REPLACEMENT POLICY

- What if:
  - We could predict the future (… with a magical oracle)
  - All future page accesses are known
  - Always replace the page in the cache used farthest in the future

- Used for a comparison
- Provides a "best case" replacement policy

- Consider a 3-element empty cache with the following page accesses:

0 1 2 0 1 3 0 3 1 2 1   **What is the hit/miss ratio?**

6 hits

FIFO REPLACEMENT

- Queue based
- Always replace the oldest element at the back of cache
- Simple to implement
- Doesn't consider importance... just arrival ordering

- Consider a 3-element empty cache with the following page accesses:

  0 1 2 0 1 3 0 3 1 2 1

- What is the hit/miss ratio?   **4 hits**
- How is FIFO different than LRU?   **LRU incorporates history**

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.16

RANDOM REPLACEMENT

- Pick a page at random to replace
- Simple and fast implementation
- Performance depends on luck of random choices

  0 1 2 0 1 3 0 3 1 2 1

Random Performance over 10,000 Trials

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.17

HISTORY-BASED POLICIES

- LRU: Least recently used
- Always replace page with oldest access time (front)
- Always move end of cache when element is read again
- Considers temporal locality (*when pg was last accessed*)

  0 1 2 0 1 3 0 3 1 2 1   **What is the hit/miss ratio?**
                                         6 hits

- LFU: Least frequently used
- Always replace page with fewest accesses (front)
- Consider frequency of page accesses

  0 1 2 0 1 3 0 3 1 2 1   **Hit/miss ratio is=**
                                         6 hits

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.18

WORKLOAD EXAMPLES: NO-LOCALITY

- No-Locality (Random Access) Workload
  - Perform 10,000 random page accesses
  - Across set of 100 memory pages

When the cache is large enough to fit the entire workload, it doesn't matter which policy you use.

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.19

WORKLOAD EXAMPLES: 80/20

- 80/20 Workload
  - Perform 10,000 page accesses, against set of 100 pages
  - 80% of accesses are to 20% of pages (hot pages)
  - 20% of accesses are to 80% of pages (cold pages)

LRU is more likely to hold onto hot pages

(recalls history)

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.20

WORKLOAD EXAMPLES: SEQUENTIAL

- Looping sequential workload
  - Refer to 50 pages in sequence: 0, 1, …, 49
  - Repeat loop

Random performs better than FIFO and LRU for cache sizes < 50

Algorithms should provide "scan resistance"

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.21

IMPLEMENTING LRU

- Implementing last recently used (LRU) requires tracking access time for all system memory pages
- Times can be tracked with a list
- For cache eviction, we must scan an entire list
- Consider: 4GB memory system (2^{32}),
  with 4KB pages (2^{12})

- This requires 2^{20} comparisons !!!

- Simplification is needed
  - Consider how to approximate the oldest page access

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.22

IMPLEMENTING LRU - 2
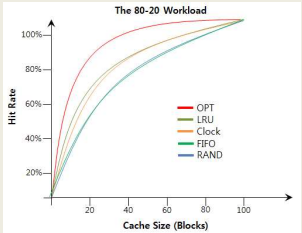
- Harness the Page Table Entry (PTE) Use Bit
- HW sets to 1 when page is used
- OS sets to 0

- Clock algorithm (*approximate LRU*)
  - Refer to pages in a circular list
  - Clock hand points to current page
  - Loops around
    - IF USE_BIT=1 set to USE_BIT = 0
    - IF USE_BIT=0 replace page

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.23

CLOCK ALGORITHM

- Not as efficient as LRU, but better than other replacement algorithms that do not consider history

The 80-20 Workload

Hit Rate

OPT
LRU
Clock
FIFO
RAND

Cache Size (Blocks)

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.24

CLOCK ALGORITHM - 2

- Consider dirty pages in cache
- If DIRTY (modified) bit is FALSE
  - No cost to evict page from cache

- If DIRTY (modified) bit is TRUE
  - Cache eviction requires updating memory
  - Contents have changed

- Clock algorithm should favor no cost eviction

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.25

WHEN TO LOAD PAGES

- On demand → demand paging

- Prefetching
  - Preload pages based on anticipated demand

  - Prediction based on locality
  - Access page P, suggest page P+1 may be used

- What other techniques might help anticipate required memory pages?
  - Prediction models, historical analysis
  - In general: accuracy vs. effort tradeoff
  - High analysis techniques struggle to respond in real time

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.26

OTHER SWAPPING POLICIES

- Page swaps / writes
  - Group/cluster pages together
  - Collect pending writes, perform as batch
  - Grouping disk writes helps amortize latency costs

- Thrashing
  - Occurs when system runs many memory intensive processes and is low in memory
  - Everything is constantly swapped to-and-from disk

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.27

OTHER SWAPPING POLICIES - 2

- Working sets
  - Groups of related processes
  - When thrashing: prevent one or more working set(s) from running
  - Temporarily reduces memory burden
  - Allows some processes to run, reduces thrashing

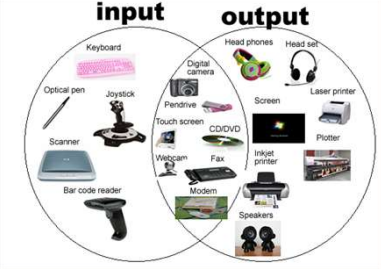| December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.28 |

CHAPTER 36:
I/O DEVICES

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.29 |

OBJECTIVES

- Chapter 36

  - Polling vs Interrupts

  - Programmed I/O (PIO)
    - Port-mapped I/O (PMIO)
    - Memory-mapped I/O (MMIO)

  - Direct memory Access (DMA)

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.30 |

I/O DEVICES

- Modern computer systems interact with a variety of devices

input   output

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
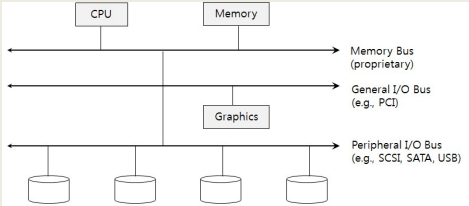School of Engineering and Technology, University of Washington - Tacoma   L18.31

COMPUTER SYSTEM ARCHITECTURE

Prototypical System Architecture

VERY FAST: CPU is attached to main memory via a **Memory bus**
FAST: High speed devices (e.g. video) are connected via a **General I/O bus**
SLOWER: Disks are connected via a **Peripheral I/O bus**

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.32

I/O BUSES

- Buses
  - Buses closer to the CPU are faster
  - Can support fewer devices
  - Further buses are slower, but support more devices

- Physics and costs dictate "levels"
  - Memory bus
  - General I/O bus
  - Peripheral I/O bus

- Tradeoff space: speed vs. locality

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.33

CANONICAL DEVICE

- Consider an arbitrary canonical *"standard/generic"* device

Registers:	Status	Command	Data	interface

Micro-controller(CPU)
Memory (DRAM or SRAM or both)
Other Hardware-specific Chips   internals

Canonical Device

- Two primary components
  - Interface (registers for communication)
  - Internals: Local CPU, memory, specific chips, firmware (embedded software)

December 3, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L18.34

CANONICAL DEVICE:
HARDWARE INTERFACE

- Status register
  - Maintains current device status

- Command register
  - Where commands for interaction are sent

- Data register
  - Used to send and receive data to the device

General concept:
The OS interacts and controls device behavior
by reading and writing the device registers.

December 3, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L18.35

OS DEVICE INTERACTION

- Common example of device interaction

```
while ( STATUS == BUSY)        Poll- Is device available?
    ; //wait until device is not busy
write data to data register        Command parameterization
write command to command register   Send command
    Doing so starts the device and executes the command
while ( STATUS == BUSY)        Poll – Is device done?
    ; //wait until device is done with your request
```

December 3, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L18.36

POLLING

- OS checks if device is *READY* by repeatedly checking the STATUS register
  - Simple approach
  - CPU cycles are wasted without doing meaningful work
  - Ok if only a few cycles, for rapid devices that are often READY
  - BUT polling, as with "spin locks" we understand is inefficient

CPU utilization by polling

INTERRUPTS VS POLLING

- For longer waits, put process waiting on I/O to sleep
- Context switch (C/S) to another process
- When I/O completes, fire an interrupt to initiate C/S back
  - Advantage: better multi-tasking and CPU utilization
  - Avoids: unproductive CPU cycles (polling)

Diagram of CPU utilization by interrupt

INTERRUPTS VS POLLING - 2

What is the tradeoff space ?

- Interrupts are not always the best solution

  - How long does the device I/O require?

  - What is the cost of context switching?

> If device I/O is fast → **polling** is better.
> If device I/O is slow → **interrupts** are better.

INTERRUPTS VS POLLING - 3

- One solution is a two-phase hybrid approach
  - Initially poll, then sleep and use interrupts

- Livelock problem
  - Common with network I/O
  - Many arriving packets generate *many many* interrupts
  - Overloads the CPU!
  - No time to execute code, just interrupt handlers !

- Livelock optimization
  - Coalesce multiple arriving packets (for different processes) into fewer interrupts
  - Must consider number of interrupts a device could generate

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.40 |

DEVICE I/O

- To interact with a device we must send/receive DATA

- There are two general approaches:

  - Programmed I/O (PIO)

  - Direct memory access (DMA)

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.41 |

Transfer Modes

Mode	#	Maximum transfer rate (MB/s)	cycle time
PIO	0	3.3	600 ns
	1	5.2	383 ns
	2	8.3	240 ns
	3	11.1	180 ns
	4	16.7	120 ns
Single-word DMA	0	2.1	960 ns
	1	4.2	480 ns
	2	8.3	240 ns
Multi-word DMA	0	4.2	480 ns
	1	13.3	150 ns
	2	16.7	120 ns
	3[34]	20	100 ns
	4[34]	25	80 ns
Ultra DMA	0	16.7	240 ns ÷ 2
	1	25.0	160 ns ÷ 2
	2 (Ultra ATA/33)	33.3	120 ns ÷ 2
	3	44.4	90 ns ÷ 2
	4 (Ultra ATA/66)	66.7	60 ns ÷ 2
	5 (Ultra ATA/100)	100	40 ns ÷ 2
	6 (Ultra ATA/133)	133	30 ns ÷ 2
	7 (Ultra ATA/167)[35]	167	24 ns ÷ 2

From https://en.wikipedia.org/wiki/Parallel_ATA

PROGRAMMED I/O (PIO)

- Spend CPU time to perform I/O
- CPU is involved with the data movement (input/output)
- PIO is slow –CPU is occupied with meaningless work

Diagram of CPU utilization

PIO DEVICES

- Legacy serial ports

- Legacy parallel ports

- PS/2 keyboard and mouse

- Legacy MIDI, joysticks

- Old network interfaces

PROGRAMMED I/O DEVICE (PIO) INTERACTION

- Two primary PIO methods

  - Port mapped I/O (PMIO)

  - Memory mapped I/O (MMIO)

PORT MAPPED I/O (PMIO)

- Device specific CPU I/O Instructions

- Follows a CISC model: extra instructions

- x86-x86-64: `in` and `out` instructions
- `outb`, `outw`, `outl`
- 1, 2, 4 byte copy from EAX → device's I/O port

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.46

MEMORY MAPPED I/O (MMIO)

- Device's memory is mapped to CPU memory
- Tenet of RISC CPUs: instructions are eliminated, CPU is simpler
- Old days: 16-bit CPUs didn't have a lot of spare memory space
- Today's CPUs: 32-bit (4GB addr space) & 64-bit (128 TB addr space)
- Regular CPU instructions used to access device: mapped to memory
- Devices monitor CPU address bus and respond to their addresses
- I/O device address areas of memory are **reserved** for I/O
  - Must not be available for normal memory operations.

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.47

DIRECT MEMORY ACCESS (DMA)

- Copy data in memory by **_offloading_** to "DMA controller"
- Many devices (including CPUs) integrate DMA controllers
- CPU gives DMA: memory address, size, and copy instruction
- DMA performs I/O independent of the CPU
- DMA controller generates CPU interrupt when I/O completes

Diagram of CPU utilization by DMA

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.48

DIRECTORY MEMORY ACCESS – 2

- **Many devices use DMA**
  - **HDD/SSD controllers (ISA/PCI)**
  - **Graphics cards**
  - **Network cards**
  - **Sound cards**
  - **Intra-chip memory transfer for multi-core processors**

- **DMA allows computation and data transfer time to proceed in parallel**

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.49 |

DEVICE INTERACTION

- **The OS must interact with a variety of devices**

- **Example: for DISK I/O consider the variety of disks:**

- **SCSI, IDE, USB flash drive, DVD, etc.**

- **Device drivers use abstraction to provide general interfaces for vendor specific hardware**

- **In Linux: block devices**

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.50 |

FILE SYSTEM ABSTRACTION

- **Layers of I/O abstraction in Linux**
- **C functions (open, read, write) issue block read and write requests to the generic block layer**

Application	user
POSIX API [open, read, write, close, etc]	
File System	kernel
Generic Block Interface [block read/write]	
Generic Block Layer	
Specific Block Interface [protocol-specific read/write]	
Device Driver [SCSI, ATA, etc]	

The File System Stack

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.51 |

FILE SYSTEM ABSTRACTION ISSUES

- **Too much abstraction**

- Many devices provide special capabilities
- Example: SCSI Error handling
- SCSI devices provide extra detail which are lost to the OS

- **Buggy device drivers**

- 70% of OS code is in device drivers
- Device drivers are required for every device plugged in
- Drivers are often 3rd party, which is not quality controlled at the same level as the OS (Linux, Windows, MacOS, etc.)

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.52

CH. 37: HARD DISK DRIVES

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.5 3

OBJECTIVES

- **Chapter 37**

  - **HDD Internals**

  - **Seek time**

  - **Rotational latency**

  - **Transfer speed**

  - **Capacity**

  - **Scheduling algorithms**

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.54

HARD DISK DRIVE (HDD)

- Primary means of data storage (persistence) for decades

- Consists of a large number of data **sectors**
- Sector size is 512-bytes

- An n sector HDD
  can be is addressed as an array of 0..n-1 sectors
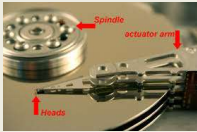
HDD INTERFACE

- Writing disk sectors is atomic (512 bytes)

- Sector writes are completely successful, or fail

- Many file systems will read/write 4KB at a time
  - Linux ext3/4 default filesystem blocksize – 4096

- Same as typical memory page size

BLOCK SIZE IN LINUX EXT4

- mkefs.ext4 -i bytes-per-inode

Specify the bytes/inode ratio. mke2fs creates an inode for every bytes-per-inode bytes of space on the disk. The larger the bytes-per-inode ratio, the fewer inodes will be created. This value generally shouldn't be smaller than the blocksize of the filesystem, since in that case more inodes would be made than can ever be used. Be warned that it is not possible to expand the number of inodes on a filesystem after it is created, so be careful deciding the correct value for this parameter.

EXAMPLE: USDA SOIL EROSION MODEL WEB SERVICE (RUSLE2)

- Host ~2,000,000 files totaling 9.5 GB on a ~20GB filesystem on a cloud-based Virtual Machine

- With default inode ratio (4096 block size), only ~488,000 files will fit

- Drive less than half full, but files will not fit !

- HDDs support a minimum block size of 512 bytes

- OS filesystems such as ext3/ext4 can support "finer grained" management at the expense of a larger catalog size

| December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.58 |

EXAMPLE: USDA SOIL EROSION MODEL WEB SERVICE (RUSLE2) - 2

- Free space in bytes (df)

Device	total size	bytes-used	bytes-free	usage
/dev/vda2	13315844	9556412	3049188	76% /mnt

- Free inodes (df –i) @ 512 bytes / node

Device	total inodes	used	free	usage
/dev/vda2	3552528	1999823	1552705	57% /mnt

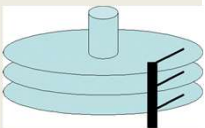| December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.59 |

HDD INTERFACE - 2

- Torn write
  - When OS uses larger block size than HDD
  - Block writes not atomic - they SPAN multiple HDD sectores
  - Upon power failure only a portion of the OS block is written

- HDD access
  - Sequential reads of sectors is fastest
  - Random sector reads are slow
  - Disk head continuously must jump to different tracks

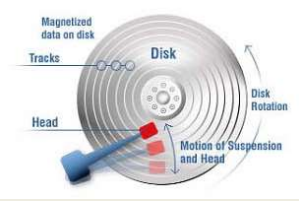| December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.60 |

HDD PLATTER

- Made from aluminum coated with thin magnetic layer
- HDD records on both sides of each platter
- Data is stored by inducing magnetic changes

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
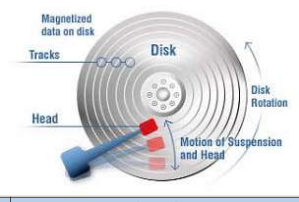School of Engineering and Technology, University of Washington - Tacoma   L18.61

HDD SPINDLE

- Connected to motor which spins the disk
- Speed measures in RPM (rotations per minute)
- Typical: 7200-15000 rpm
- 10000 rpm – 1 rotation in 6ms; 15k rpm 1 rotation in 4ms

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
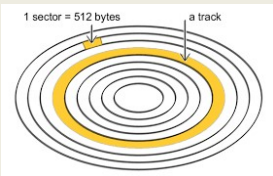School of Engineering and Technology, University of Washington - Tacoma   L18.62

HDD TRACK

- Concentric circle of sectors
- Single side of platter contains 290 K tracks (2008)
- Zones: groups of tracks with same # of sectors

Outer tracks have
More sectors

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.63

EXAMPLE: SIMPLE DISK DRIVE

- Single track disk
- Head: one per surface of drive
- Arm: moves heads across surface of platters

Rotates this way

head 6   spindle

A Single Track Plus A Head

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.64

HARD DISK STRUCTURE

track *t* — spindle

sector *s*

arm assembly

cylinder *c*

read-write head

platter

arm

rotation

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.65

SINGLE-TRACK LATENCY: THE ROTATIONAL DELAY

- Rotational latency ($T_{rotation}$): time to rotate to desired sector

- Average $T_{rotation}$ is ~ half the time of a full rotation

- Calculate time for 1 rotation based on rpm

- 7200rpm = 8.33ms per rotation = ~4.166ms
- 10000rpm = 6ms per rotation = ~3ms
- 15000rpm = 4ms per rotation = ~2ms

Rotates this way

head 6   spindle

A Single Track Plus A Head

December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.66

SEEK TIME

Three Tracks Plus A Head (Right: With Seek)
(e.g., read to sector 11)

- Seek time (T_{seek}): time to move disk arm to proper track
- Most time consuming HDD operation

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.67

FOUR PHASES OF SEEK

- Acceleration → coasting → deceleration → settling

- **Acceleration**: the arm gets moving

- **Coasting**: arm moving at full speed

- **Deceleration**: arm slow down

- **Settling**: Head is carefully positioned over track
  - Settling time is often high, from .5 to 2ms

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.68

HDD I/O

- Data transfer
  - Final phase of I/O: time to read or write to disk surface

- Complete I/O cycle:
  1. Seek (accelerate, coast, decelerate, settle)
  2. Wait on rotational latency
  3. Data transfer

December 3, 2018   TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma   L18.69

TRACK SKEW

- Sectors are offset across tracks to allow time for head to reposition for sequential reads

- Without track skew, when head is repositioned sector would have already been passed

Three Tracks: Track Skew Of 2

December 3, 2018 — TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma — L18.70

TRACK SKEW - 2

www.hddscan.com

December 3, 2018 — TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma — L18.71

HDD CACHE

- Buffer to support caching reads and writes
- Improves drive response time
- Up to 128 MB, slowly have been growing

- Two styles
  - Writeback cache
    - Report write complete immediately when data is transferred to HDD cache
    - Dangerous

  - Writethrough cache
    - Reports write complete only when write is physically completed on disk

December 3, 2018 — TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma — L18.72

TRANSFER SPEED

- I/O Time $T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$

- The rate of I/O $R_{I/O} = \dfrac{Size_{Transfer}}{T_{I/O}}$

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects Via	SCSI	SATA

Disk Drive Specs: SCSI Versus SATA

December 3, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L18.73

I/O SPEED

- **Random workload: 4KB random read on HDD**
- **Sequential workload: read 100MB contiguous sectors**

		Cheetah 15K.5	Barracuda
T_{seek}		4 ms	9 ms
$T_{rotation}$		2 ms	4.2 ms
Random	$T_{transfer}$	30 microsecs	38 microsecs
	$T_{I/O}$	6 ms	13.2 ms
	$R_{I/O}$	0.66 MB/s	0.31 MB/s
Sequential	$T_{transfer}$	800 ms	950 ms
	$T_{I/O}$	806 ms	963.2 ms
	$R_{I/O}$	125 MB/s	105 MB/s

Disk Drive Performance: SCSI Versus SATA

> There is a huge gap in drive throughput
> between random and sequential workloads

December 3, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L18.74

MODERN HDD SPECS

- **See sample HDD configurations here:**

- **https://www.hgst.com/products/hard-drives**

December 3, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L18.75

DISK SCHEDULING

- Disk scheduler: determine how to order I/O requests

- Multiple levels - OS and HW

- OS: provides ordering

- HW: further optimizes using intricate details of physical HDD implementation and state

SSTF – SHORTEST SEEK TIME FIRST

- Disk scheduling – which I/O request to schedule next
- Shortest Seek Time First (SSTF)
- Order queue of I/O requests by nearest track

SSTF: Scheduling Request 21 and 2
Issue the request to 21 → issue the request to 2

SSTF ISSUES

- Problem 1: HDD abstraction

- Drive geometry not available to OS. Nearest-block-first is a comparable alternate algorithm.

- Problem 2: Starvation

- Steady stream of requests for local tracks may prevent arm from traversing to other side of platter

DISK SCHEDULING ALGORITHMS

- **SWEEP**
- Single repeated passes across disk
- Issue: if request arrives for a recently visited track it will not be revisited until a full cycle completes

- **F-SCAN**
- Freeze request queue during sweep
- Cache arriving requests until later

- **Elevator (C-SCAN)** – circular scan
- Sweep from outer to inner track and reverse, inner to outer track, etc.

December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.79

SHORTEST TIME POSITIONING FIRST

- Determine next sector to read?

- On which track?

- On which sector?

On modern drives, both seek and rotation are roughly equivalent: **Thus, SPTF (Shortest Positioning Time First) is useful.**

December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.80

I/O MERGING
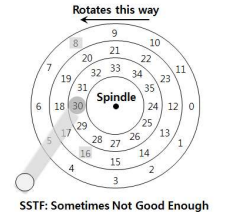
- Group temporary adjacent requests
- Reduce overhead
- Read (memory blocks): 33 8 34

- How long we should wait for I/O ?

- When do we know we have waited too long?

December 3, 2018 | TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma | L18.81

QUESTIONS

Magnetic Recording System Technologies

New recording system technologies are needed to keep the HDD industry on its historical track of delivering capacity improvements over time

Bit Patterned Magnetic Recording (BPMR)
Heat Assisted Magnetic Recording (HAMR)
Microwave Assisted Magnetic Recording (MAMR)

Future recording technologies will build on SMR, not replace it.

Shingled Magnetic Recording (SMR)

Discrete Track Recording (DTR)
(Limited Gains)

Perpendicular Magnetic Recording (PMR)

Longitudinal Magnetic Recording (LMR)

10 Tb/in^2
5 Tb/in^2
1 Tb/in^2
Superpara-magnetic limit
150 Gb/in^2

Areal Density

Time

Y. Shiroishi, Intermag 2009, FA-01

HDD CAPACITY

- Superparamagnetism limits HDD capacity

- In sufficiently small nanoparticles, magnetization can randomly flip direction under the influence of temperature.

- HDD capacity is limited by the minimum usable size of particles – the superparamagnetic limit.

HDD CAPACITY - 2

- Longitudinal recording: 100-200GB/in
- Perpendicular recording: 667 GB/in
- Future technologies under development

"Ring" writing element

Longitudal Recording (standard)

Recording layer

"Monopole" writing element

Perpendicular Recording

Recording Layer
Additional Layer

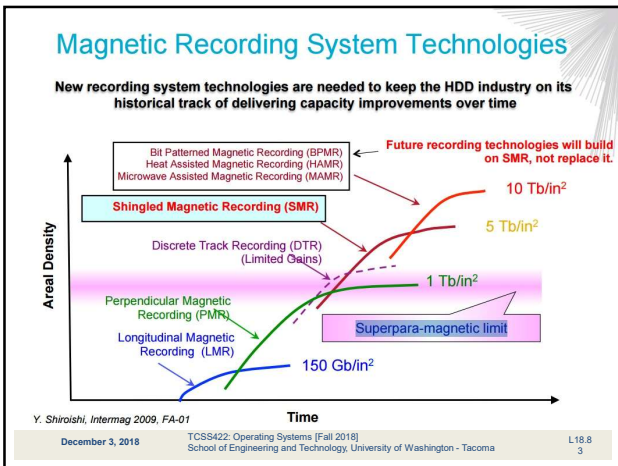| December 3, 2018 | TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma | L18.85 |