Part1

Monday, May 13, 2019 12:00 PM

Device Controllers

- I/O devices have components
  - Mechanical components
  - Electronic components
- Electronic component controls the device
  - May be able to handle multiple devices
  - May be more than one controller per mechanical component
  - Such as hard drive
- Controller's tasks
  - Convert serial bit stream to block of bytes
  - Perform error correction as necessary
  - Make available to main memory

How is memory mapped I/O done?

- Single bus
  - All memory accesses go over a shared bus
  - I/O and RAM accesses compete for bandwidth
- Dual Bus
  - RAM access over high speed bus
  - I/O access over low speed bus
  - Less competition for bus
  - More hardware (more expensive)

I/O software: goals

- Device independence
  - Programs can access any I/O device
  - No need to specify device in advance
- Uniform naming
  - Name of a file or device is a string or an integer
  - Doesn't depend on the machine (underlying hardware)
- Error handling
  - Done as close to the hardware as possible
  - Isolate higher-level software
- Synchronous vs asynchronous transfers
  - Blocked transfers vs interrupt-driven
- Buffering
  - Date coming off a device cannot be stored in final destination
- Sharable vs dedicated devices

Interrupt handlers

- Interrupt handlers are best hidden
  - Driver starts an I/O operation and blocks
  - Interrupt notifies of completion
- Interrupt procedure does its task
  - Then unblocks driver that started it
  - Perform minimal actions at interrupt time
    - Some of the functionality can be done by the driver after it is unblocked

- Interrupt handler must
    - Save registers not already save by interrupt hardware
    - Set up context for interrupt service procedure

Device drivers

- Device drivers go between device controllers and rest of OS
    - Drivers standardize interface to widely varied devices
- Device drivers communicate with controllers over bus
    - Controllers communicate with devices themselves

Disk drive structure

- Data stored on surfaces
    - Up to two surfaces per platter
    - One or more platters per disk
- Data in concentric tracks
    - Tracks broken into sectors
        - 256B - 4KB per sector
    - Cylinder: corresponding tracks on all surfaces
- Data read and written by heads
    - Actuator moves heads
    - Heads move in unison

Part2

Friday, May 17, 2019 12:04 PM

Disk Request scheduling

- Goal: use disk hardware efficiently
  - Bandwidth as high as possible
  - Disk transferring as often as possible (and not seeking)
- We want to
  - Minimize disk seek time (moving from track to track)
  - Minimize rotational latency (waiting for disk to rotate the desired sector under the read/write head)
- Calculate disk bandwidth by
  - Total bytes transferred / time to service request
  - Seek time and rotational latency are overhead (no data is transferred), and reduce disk bandwidth

Disk scheduling algorithms

- Schedule disk requests to minimize disk seek time
  - Seek time increases as distance increases (though not linearly)

First Come First serve

- Requests serviced in the order in which they arrived
  - Easy to implement
  - May involve lots of unnecessary seek distance
- Seek order
  - 100 175 51 133 8 140 73 77
  - Start 63
  - Total cylinders = 646

Shortest seek time first

- Service the request with the shortest seek time from the current head position
  - Form of SJF scheduling
  - May starve some requests
- Seek order
  - 73 77 51 8 100 133 140 175
  - Start 63
  - Total cylinders = 250

SCAN (elevator algorithm)

- Disk arm starts at one end of the disk and moves towards the other end servicing requests as it goes
  - Reverses direction when it gets to end of the disk
  - Also known as elevator algorithm
- Seek order
  - 51 8 0 73 77 100 133 140 175
  - Start 63
  - Total cylinders = 238

C-SCAN

- Identical to SCAN except head returns to cylinder 0 when it reaches the end of the disk

    - Treats cylinder list as a circular list that wraps around the disk
    - Waiting time is more uniform for cylinders near the edge of the disk
- Seek order
    - 73 77 100 133 140 175 199 0 8 51
    - Start 63
    - Total cylinders = 386

C-LOOK

- Identical to C-SCAN except head only travels as far as the last request in each direction
    - Saves seek time from last sector to end of disk
- Seek order
    - 73 77 100 133 140 175 8 51
    - Start 63
    - Total cylinders = 322

Picking a disk scheduling algorithm

- SSTF is easy to implement and works OK if there aren't too many disk requests in the queue
- SCAN-type algorithms perform better for systems under heavy load
    - More fair than SSTF
    - Use LOOK rather than SCAN algorithms to save time
- Long seeks aren't too expensive, so choose C-LOOK over LOOK to make response time more even
- Disk request scheduling interacts with algorithms for allocating blocks to files
    - Make scheduling algorithm modular: allow it to be changed without changing the file system
- Use SSTF or FCFS for lightly loaded systems
- Use C-LOOK for heavily loaded systems

Flash memory and SSDs

- Compared to disk, flash is
    - Faster (shorter access time, but lower bandwidth
    - More expensive
    - More reliable (devices)
    - Less reliable (sectors)
- Compared to DRAM, flash is
    - Cheaper (a bit)
    - Non-volatile (data survives power loss)
    - Slower
- Use flash as a level between disk and memory?
- Issues
    - Can't overwrite sectors "in place"
    - Flash wears out: can only write 3,000 to 100,000 times per memory cell

Writing to flash memory

- Can't overwrite page in place
    - Need to write to "clean" region
    - Unit of erase is "erase block"
- One solution: copy entire erase block to new location
    - Modify the page being written
- This isn't efficient!
    - Flash vendors spend a lot of time on Flash Translation layers
    - FTLs map logical page numbers to physical pages
    - Also manage wear leveling: distribute erases among blocks

FTL lies. It makes it look like a disk

Is there a better way to manage flash

- Log-structured data storage works well on flash
    - Append updates to a "log"
- Periodically "clean" structures
    - Rewrite live data to a new log segment
    - Obsolete data isn't rewritten
- Need to track mapping of logical locks to physical locations

Clock hardware

- Crystal oscillator with fixed frequency (only when computer is on)
    - Typically used too time short intervals (~1 second)
    - May be used to correct time-of-day clock
- Time-of-day clock (runs when system is off)
    - Keeps minutes, hours, days
    - May not be too accurate…
    - Used to load system clock at startup
- Time kept in seconds and ticks (often 10^2 to 10^6 per second)
    - Number of seconds since a particular time
        - For many versions of Unix, tick 0 was on January 1, 1970
    - Number of ticks this second
    - Advance ticks once per clock interrupt
    - Advance seconds when ticks "overflow"

Use timers in software

-