Persistence

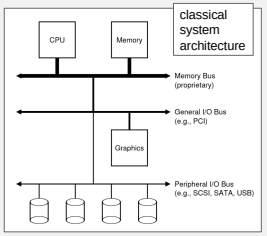
Piece #3

- virtualization – CPU, memory
- concurrency
- **persistence**

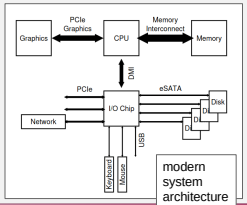
- the contents of memory are lost when the power is shut off, so *persistent storage* is needed
- this involves I/O devices

System Architecture

- system architecture is hierarchical
  - connections must be short to be fast
    - not much room for devices on a high-performance bus
  - high performance costs more



classical system architecture

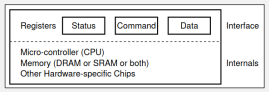


modern system architecture (example)

Devices

a typical (simplified) view

- a device has two main components
  - hardware interface
  - internal structure



- hardware interface
  - *status* register – to read device's current status
  - *command* register – to pass a command to the device
  - *data* register – to pass or get data
- basic protocol

```
While (STATUS == BUSY)
    ; // wait until device is not busy
Write data to DATA register
Write command to COMMAND register
    (starts the device and executes the command)
While (STATUS == BUSY)
    ; // wait until device is done with your request
```

  - called *programmed I/O* (PIO) when the CPU is involved in data transfer

Improving Performance

- polling is inefficient for slow devices
- instead, the process blocks and the device raises a *hardware interrupt* when it is done
  - allows other processes to run in the interim
  - allows for overlap of computation and I/O
- interrupts are not always better
  - context switches take time
  - a continuous stream of interrupts can result in the OS doing nothing else (no user processes run)

Improving Performance

Strategies –

- can employ a two-phased approach if device speeds are unknown or vary
  - poll for a little while, then block and wait for the interrupt if the device hasn't finished yet
- occasionally poll instead of blocking so that interrupts aren't generated and other processes can run
- *coalesce* interrupts
  - device waits before raising an interrupt, so that multiple request-completions can be handled at once
  - tradeoff between lowering overhead of interrupt handling and increasing latency

Improving Performance

- with PIO, the CPU must copy the data being written to disk from memory one word at a time
- *direct memory access* (DMA) shifts the copying work to the DMA controller and away from the CPU

Device Interaction

- explicit I/O instructions
  - privileged hardware instructions allowing access to the device registers
- memory-mapped I/O
  - device registers are available as memory locations
  - OS issues load or store operations; hardware routes to device

Device Drivers

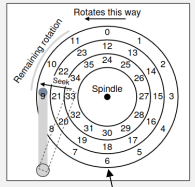
- *device drivers* provide an abstraction hiding the particulars of specific devices
  - the vast array of devices means that device drivers end up being a huge percentage of the OS code

Hard Disks

- hard disks have been the main form of persistent storage device
  - first introduced by IBM in 1957
- the view of the disk is an array of *n sectors*
  - traditionally 512 byte blocks, though Advanced Format (4096 byte blocks) has been standard for 15+ years
  - only single-sector writes are guaranteed to be atomic, though reads/writes may be clustered into larger blocks

Hard Disks

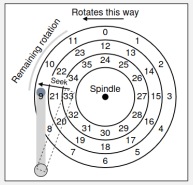
- geometry
  - one or more *platters*
  - each platter has two *surfaces*
  - each surface is divided into concentric *tracks*
  - a *disk head* attached to a *disk arm* reads/writes
    - one disk head per surface
  - a motor drives the central *spindle* to rotate the platters
    - typically 7,200-15,000 RPM (4-8ms per rotation)



https://en.wikipedia.org/wiki/Hard_disk_drive#/media/File:Hatachi_500_GB_hard_drive,_2011.jpg

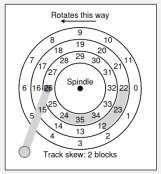
Hard Disks

- hard disk access time
  - *seek time* for the head to move to the correct track
  - *rotational delay* for the desired sector to rotate under the head
  - *transfer time* for the data to be read or written



Hard Disks

- other details
  - *track skew* is employed to account for the time to reposition the head when the next sector is in a different track



  - outer tracks have more sectors than inner tracks, so *multi-zoned drives* group tracks into zones with the same number of sectors per track
  - *track buffer* holds data to be read/written
    - allows additional sectors to be read, improving performance of subsequent requests for nearby sectors
    - *write back* caching means the drive reports completion when data is in the buffer
    - *write through* caching means the drive reports completion only after data is written to the disk

Numbers – Key Points

- there is a huge difference in performance between random and sequential disk accesses
  - transfer data sequentially if possible, and in as large chunks as possible otherwise

	Cheetah	Barracuda
$R_{I/O}$ Random	0.66 MB/s	0.31 MB/s
$R_{I/O}$ Sequential	125 MB/s	105 MB/s

- there is a substantial difference in performance between disks built for performance and those built for cheap capacity