I/O devices can be characterized by:
- Behavior: input, output, storage
- Partner: human or machine
- Data rate: bytes/sec, transfers/sec

I/O bus connections

Typical x86 PC I/O System

Disk Storage

• Nonvolatile, rotating magnetic storage
Disk Drive Terminology

- Data is recorded on concentric tracks on both sides of a platter
  - Tracks are organized as fixed size (bytes) sectors
- Corresponding tracks on all platters form a cylinder
- Data is addressed by three coordinates: cylinder, platter, and sector

Disk Sectors and Access

- Each sector records
  - Sector ID
  - Data (512 bytes, 4096 bytes proposed)
  - Error correcting code (ECC)
    - Used to hide defects and recording errors
    - Synchronization fields and gaps
- Access to a sector involves
  - Queuing delay if other accesses are pending
  - Seek: move the heads
  - Rotational latency
  - Data transfer
  - Controller overhead

Disk Performance

- Actuator moves (seek) the correct read/write head over the correct sector
  - Under the control of the controller
- Disk latency = controller overhead + seek time + rotational delay + transfer delay
  - Seek time and rotational delay are limited by mechanical parts

Disk Performance

- Seek time determined by the current position of the head, i.e., what track it covering, and the new position of the head
  - millisecond
- Average rotational delay is time for 0.5 revolutions
- Transfer rate is a function of bit density
Disk Access Example

- Given
  - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk
- Average read time
  - 4ms seek time + \( \frac{1}{2} \times \frac{15,000}{60} \approx 2\) ms rotational latency + \( \frac{512}{100MB/s} = 0.005\) ms transfer time + 0.2ms controller delay = 6.2ms
- If actual average seek time is 1ms
  - Average read time = 3.2ms

Disk Performance Issues

- Manufacturers quote average seek time
  - Based on all possible seeks
  - Locality and OS scheduling lead to smaller actual average seek times
- Smart disk controller allocate physical sectors on disk
  - Present logical sector interface to host
  - Standards: SCSI, ATA, SATA
- Disk drives include caches
  - Prefetch sectors in anticipation of access
  - Avoid seek and rotational delay
  - Maintain caches in host DRAM

Arrays of Inexpensive Disks: Throughput

- Data is striped across all disks
- Visible performance overhead of drive mechanics is amortized across multiple accesses
- Scientific workloads are well suited to such organizations

Arrays of Inexpensive Disks: Request Rate

- Consider multiple read requests for small blocks of data
- Several I/O requests can be serviced concurrently
Reliability of Disk Arrays

- The reliability of an array of N disks is lower than the reliability of a single disk
  - Any single disk failure will cause the array to fail
  - The array is N times more likely to fail
- Use redundant disks to recover from failures
  - Similar to use of error correcting codes
- Overhead
  - Bandwidth and cost

RAID

- Redundant Array of Inexpensive (Independent) Disks
  - Use multiple smaller disks (c.f. one large disk)
  - Parallelism improves performance
  - Plus extra disk(s) for redundant data storage
- Provides fault tolerant storage system
  - Especially if failed disks can be “hot swapped”

RAID Level 0

- RAID 0 corresponds to use of striping with no redundancy
- Provides the highest performance
- Provides the lowest reliability
- Frequently used in scientific and supercomputing applications where data throughput is important

RAID Level 1

- The disk array is “mirrored” or “shadowed” in its entirety
- Reads can be optimized
  - Pick the array with smaller queuing and seek times
- Performance sacrifice on writes – to both arrays
RAID 3: Bit-Interleaved Parity

- N + 1 disks
  - Data striped across N disks at byte level
  - Redundant disk stores parity
  - Read access
    - Read all disks
  - Write access
    - Generate new parity and update all disks
  - On failure
    - Use parity to reconstruct missing data

- Not widely used

RAID Level 4: N+1 Disks

- Data is interleaved in blocks, referred to as the striping unit and striping width
- Small reads can access subset of the disks
- A write to a single disk requires 4 accesses
  - read old block, write new block, read and write parity disk
- Parity disk can become a bottleneck

The Small Write Problem

- Two disk read operations followed by two disk write operations

RAID 5: Distributed Parity

- N + 1 disks
  - Like RAID 4, but parity blocks distributed across disks
    - Avoids parity disk being a bottleneck

- Widely used
RAID Summary

- RAID can improve performance and availability
  - High availability requires hot swapping
- Assumes independent disk failures
  - Too bad if the building burns down!
- See “Hard Disk Performance, Quality and Reliability”

Flash Storage

- Nonvolatile semiconductor storage
  - 100× – 1000× faster than disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)

Flash Types

- NOR flash: bit cell like a NOR gate
  - Random read/write access
  - Used for instruction memory in embedded systems
- NAND flash: bit cell like a NAND gate
  - Denser (bits/area), but block-at-a-time access
  - Cheaper per GB
  - Used for USB keys, media storage, ...
- Flash bits wears out after 1000’ s of accesses
  - Not suitable for direct RAM or disk replacement
  - Wear leveling: remap data to less used blocks

Solid State Disks

- Replace mechanical drives with solid state drives
- Superior access performance
- Adding another level to the memory hierarchy
  - Disk is the new tape!
- Wear-leveling management

Wikipedia: PCIe DRAM and SSD
Fusion-I/O
Interconnecting Components

- Need interconnections between CPU, memory, I/O controllers
- Bus: shared communication channel
  - Parallel set of wires for data and synchronization of data transfer
  - Can become a bottleneck
- Performance limited by physical factors
  - Wire length, number of connections
- More recent alternative: high-speed serial connections with switches
  - Like networks
- What do we want
  - Processor independence, control, buffered isolation

Bus Types

- Processor-Memory buses
  - Short, high speed
  - Design is matched to memory organization
- I/O buses
  - Longer, allowing multiple connections
  - Specified by standards for interoperability
  - Connect to processor-memory bus through a bridge

Bus Signals and Synchronization

- Data lines
  - Carry address and data
  - Multiplexed or separate
- Control lines
  - Indicate data type, synchronize transactions
- Synchronous
  - Uses a bus clock
- Asynchronous
  - Uses request/acknowledge control lines for handshaking
### I/O Bus Examples

<table>
<thead>
<tr>
<th>Bus</th>
<th>Intended use</th>
<th>Devices per channel</th>
<th>Data width</th>
<th>Peak bandwidth</th>
<th>Hot pluggable</th>
<th>Max length</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewire</td>
<td>External</td>
<td>63</td>
<td>4</td>
<td>500MB/s or 100MB/s</td>
<td>Yes</td>
<td>4.5m</td>
<td>IEEE 1394</td>
</tr>
<tr>
<td>USB 2.0</td>
<td>External</td>
<td>127</td>
<td>2</td>
<td>0.2MB/s or 1.5MB/s, or 60MB/s</td>
<td>Yes</td>
<td>5m</td>
<td>USB Implementers Forum</td>
</tr>
<tr>
<td>PCI Express</td>
<td>Internal</td>
<td>1</td>
<td>3/lane</td>
<td>250MB/s or 1MB/s, or 4MB/s, or 8x, 16x, 32x</td>
<td>Depends</td>
<td>0.5m</td>
<td>PCI-SIG</td>
</tr>
<tr>
<td>Serial ATA</td>
<td>Internal</td>
<td>1</td>
<td>4</td>
<td>300MB/s or 300MB/s</td>
<td>Yes</td>
<td>1m</td>
<td>SATA-IO</td>
</tr>
<tr>
<td>Serial Attached SCSI</td>
<td>External</td>
<td>4</td>
<td>4</td>
<td>300MB/s or 300MB/s</td>
<td>Yes</td>
<td>8m</td>
<td>INCITS TC  T10</td>
</tr>
</tbody>
</table>

### PCI Express

- Standardized local bus
- Load store flat address model
- Packet based split transaction protocol
- Reliable data transfer

#### PCI Express: Operation

- Packet-based, memory mapped operation

### The Big Picture

- From electronicdesign.com
Local Interconnect Standards

- HyperTransport
  - Packet switched, point-to-point link
  - HyperTransport Consortium (AMD)

- Quickpath Interconnect
  - Packet switched, point-to-point link
  - Intel Corporation

I/O Management

- I/O is mediated by the OS
  - Multiple programs share I/O resources
    - Need protection and scheduling
  - I/O causes asynchronous interrupts
    - Same mechanism as exceptions
  - I/O programming is fiddly
    - OS provides abstractions to programs

I/O Commands

- I/O devices are managed by I/O controller hardware
  - Transfers data to/from device
  - Synchronizes operations with software

- Command registers
  - Cause device to do something

- Status registers
  - Indicate what the device is doing and occurrence of errors

- Data registers
  - Write: transfer data to a device
  - Read: transfer data from a device
I/O Register Mapping

- Memory mapped I/O
  - Registers are addressed in same space as memory
  - Address decoder distinguishes between them
  - OS uses address translation mechanism to make them only accessible to kernel
- I/O instructions
  - Separate instructions to access I/O registers
  - Can only be executed in kernel mode
  - Example: x86

Polling

- Periodically check I/O status register
  - If device ready, do operation
  - If error, take action
- Common in small or low-performance real-time embedded systems
  - Predictable timing
  - Low hardware cost
- In other systems, wastes CPU time

Interrupts

- When a device is ready or error occurs
  - Controller interrupts CPU
- Interrupt is like an exception
  - But not synchronized to instruction execution
  - Can invoke handler between instructions
  - Cause information often identifies the interrupting device
- Priority interrupts
  - Devices needing more urgent attention get higher priority
  - Can interrupt handler for a lower priority interrupt

I/O Data Transfer

- Polling and interrupt-driven I/O
  - CPU transfers data between memory and I/O data registers
  - Time consuming for high-speed devices
- Direct memory access (DMA)
  - OS provides starting address in memory
  - I/O controller transfers to/from memory autonomously
  - Controller interrupts on completion or error
Direct Memory Access

- Program the DMA engine with
  - start and destination addresses
  - Transfer count
- Interrupt-driven or polling interface
- What about use of virtual vs. physical addresses?
- Example

DMA/Cache Interaction

- If DMA writes to a memory block that is cached
  - Cached copy becomes stale
- If write-back cache has dirty block, and DMA reads memory block
  - Reads stale data
- Need to ensure cache coherence
  - Flush blocks from cache if they will be used for DMA
  - Or use non-cacheable memory locations for I/O

I/O System Design

- Satisfying latency requirements
  - For time-critical operations
  - If system is unloaded
    - Add up latency of components
- Maximizing throughput
  - Find "weakest link" (lowest-bandwidth component)
  - Configure to operate at its maximum bandwidth
  - Balance remaining components in the system
- If system is loaded, simple analysis is insufficient
  - Need to use queuing models or simulation

Measuring I/O Performance

- I/O performance depends on
  - Hardware: CPU, memory, controllers, buses
  - Software: operating system, database management system, application
  - Workload: request rates and patterns
- I/O system design can trade-off between response time and throughput
  - Measurements of throughput often done with constrained response-time
Transaction Processing Benchmarks

- Transactions
  - Small data accesses to a DBMS
  - Interested in I/O rate, not data rate

- Measure throughput
  - Subject to response time limits and failure handling
  - ACID (Atomicity, Consistency, Isolation, Durability)
  - Overall cost per transaction

- Transaction Processing Council (TPC) benchmarks (www.tcp.org)
  - TPC-APP: B2B application server and web services
  - TPC-C: on-line order entry environment
  - TPC-E: on-line transaction processing for brokerage firm
  - TPC-H: decision support — business oriented ad-hoc queries

File System & Web Benchmarks

- SPEC System File System (SFS)
  - Synthetic workload for NFS server, based on monitoring real systems
  - Results
    - Throughput (operations/sec)
    - Response time (average ms/operation)

- SPEC Web Server benchmark
  - Measures simultaneous user sessions, subject to required throughput/session
  - Three workloads: Banking, Ecommerce, and Support

I/O vs. CPU Performance

- Amdahl’s Law
  - Don’t neglect I/O performance as parallelism increases compute performance

- Example
  - Benchmark takes 90s CPU time, 10s I/O time
  - Double the number of CPUs/2 years
    - I/O unchanged

<table>
<thead>
<tr>
<th>Year</th>
<th>CPU time</th>
<th>I/O time</th>
<th>Elapsed time</th>
<th>% I/O time</th>
</tr>
</thead>
<tbody>
<tr>
<td>now</td>
<td>90s</td>
<td>10s</td>
<td>100s</td>
<td>10%</td>
</tr>
<tr>
<td>+2</td>
<td>45s</td>
<td>10s</td>
<td>55s</td>
<td>18%</td>
</tr>
<tr>
<td>+4</td>
<td>23s</td>
<td>10s</td>
<td>33s</td>
<td>31%</td>
</tr>
<tr>
<td>+6</td>
<td>11s</td>
<td>10s</td>
<td>21s</td>
<td>47%</td>
</tr>
</tbody>
</table>

I/O System Characteristics

- Dependability is important
  - Particularly for storage devices

- Performance measures
  - Latency (response time)
  - Throughput (bandwidth)
  - Desktops & embedded systems
    - Mainly interested in response time & diversity of devices
  - Servers
    - Mainly interested in throughput & expandability of devices
Dependability

- Fault: failure of a component
  - May or may not lead to system failure

Service accomplishment
  Service delivered as specified

Restoration

Failure

Service interruption
  Deviation from specified service

Dependability Measures

- Reliability: mean time to failure (MTTF)
- Service interruption: mean time to repair (MTTR)
- Mean time between failures
  - MTBF = MTTF + MTTR
- Availability = MTTF / (MTTF + MTTR)
- Improving Availability
  - Increase MTTF: fault avoidance, fault tolerance, fault forecasting
  - Reduce MTTR: improved tools and processes for diagnosis and repair

Concluding Remarks

- I/O performance measures
  - Throughput, response time
  - Dependability and cost also important
- Buses used to connect CPU, memory, I/O controllers
  - Polling, interrupts, DMA
- I/O benchmarks
  - TPC, SPECSFS, SPECWeb
- RAID
  - Improves performance and dependability

Study Guide

- Provide a step-by-step example of how each of the following work
  - Polling, DMA, interrupts, read/write accesses in a RAID configuration, memory mapped I/O
- Compute the bandwidth for data transfers to/from a disk
- Delineate and explain different types of benchmarks
- How is the I/O system of a desktop or laptop different from that of a server?
- Recognize the following standards: QPI, HyperTransport, PCIe