IO
Today

• IO
Key Points

• CPU interface and interaction with IO IO devices
• The basic structure of the IO system (north bridge, south bridge, etc.)
• The key advantages of high speed serial lines.
• The benefits of scalability and flexibility in IO interfaces
• Disks
  • Rotational delay vs seek delay
  • Disks are slow.
  • Techniques for making disks faster.
• Flash and SSDs
  • How does flash memory store bits
  • How do you turn flash memory into a usable disk?
  • Why is that hard?
IO Devices

- Large Hadron Collider: 700MB/s
- Hard drive: 50-120MB/s
- Keyboard: 10Byte/s
- 30in display: 60Hz, 1GB/s
Hooking Things to Your (Parents’) Computer

- What do we want in an IO system?
What IO Should be

- Lots of devices
  - Keyboards -- slowest
  - Printers
  - Display
  - Disks
  - Network connection
  - Digital cameras
  - Scanners
  - Scientific equipment
- Easy to hook up
  - “Plug and play”
  - The fewer wires the better.
- Easy to make sw work
  - No drivers!
  - “just works”
- Performance
  - Fast!!!!
  - Low latency
  - High bandwidth
  - Low power
- Cost
  - Cheap
  - Low hw and sw development costs
The CPUs World View

• The only IO that CPUs do is load and store
• “Programmed IO”
  • IO devices export “control registers” that drives map into the kernels address space
  • loads and stores to those addresses change the values in the control registers
  • Those address had better Write through and/or uncached
  • Fine for small scale accesses
• Direct memory access
  • The CPU is slow for moving bytes around, and it’s busy too!
  • DMA allows devices directly read and write memory
  • Fill a buffer with some data, start the DMA (via PIO), go do other things.
Interrupts

• IO devices need to get the CPUs attention
  • A DMA finishes
  • A packet arrives
  • A timer goes off

• (simplified) interrupt handling
  • CPU control transfers to the OS -- pipeline flush.
  • Like a context switch or a system call
  • Where control lands depends on the ‘interrupt vector’
  • The OS examines the system state to determine what the interrupt meant and processes it accordingly.
    • Copies data out of disk buffer or network buffer
    • Delivers signal to applications
    • etc.
Connecting Devices to Processors

- **On-chip**
  - Fastest possible connection.
  - Wide -- you can have lots of wires between devices
  - Fast -- data moves at core clock speeds
  - Cheap -- fewer chips means cheaper systems
  - Restricts flexibility -- Design is set at fab time
  - Current uses -- L2 caches, on-chip memory controller
  - Near term uses -- GPUs, network interfaces

AMD Phenom (aka barcelona)
The “Chip set”

- Off-chip is much slower.
  - Fewer wires, slower clocks (less bandwidth), and longer latency.

- North Bridge - The fast part
  - “Front side bus” in Intel-speak
  - Off-chip memory controller
  - PCI-express
  - Key system differentiator until recently.
    - Server chip sets vs desktop chip sets
    - Memory-like interface
    - Typically 64bits of data
    - Routes PIO requests to other devices
    - Lots of DMA
      - It’s sort of a data movement co-processor
    - >64GB/s of peak aggregate bandwidth
The “Chip set”

- The South bridge -- the slow part
  - Everything else...
  - USB
  - Disk IO
  - Power management
  - Real time clock
  - System status monitoring -- i2c bus
  - 100s of MB/s of bandwidth
Legacy Interfaces

• Serial lines -- RS 232
• Parallel ports
• Various expansion card interfaces
  • ISA cards
  • Nu-BUS
Legacy Disk Interfaces

- **ATA - “AT Attachment”**
  - 16 bits of data in parallel
  - 40 or 80-conductor “Ribbon cables”
  - Peak of 133MB/s
  - Two drives per cable

- **SCSI -- Small Computer System Interface**
  - Synonymous with high-end IO
  - Fast bus speeds: up to 160Mhz QDR (four data transfers per clock)
  - Many variants up to SCSI Ultra-640: 640MB/s
  - Scalable: up to 16 devices per SCSI bus.
  - Expensive.
The Serial Revolution

• Wider busses are on obvious way to increased bandwidth
  • But “jitter” and “clock skew” becomes a problem
  • If you have 32 lines in a bus, you need to wait for the slowest one.
  • All devices must use the same clock.
  • This limits bus speeds.
• Lately, high speed serial lines have been replacing wide buses.
High speed serial

• Two wires, but not power and ground
• “low voltage differential signaling”
  • If signal 1 is higher than signal 2, it’s a one
  • if signal 2 is higher, it’s a 0
  • Detecting the difference is possible at lower voltages, which further increases speed
• Max bandwidth per pair: currently 6Gb/s
• Cables are much cheaper and can be longer and cheaper -- External hard drives.
  • SCSI cables can cost $100s -- and they fail a lot.
Serial interfaces

- SATA -- Serial ATA
  - Replaces ATA
  - The logical protocol is the same, but the “transport layer” is serial instead of parallel.
  - Max performance: 300MB/s -- much less in practice.
- SAS -- Serial attached SCSI
  - Replace SCSI, Same logical protocol.
- PCIe
  - Replace PCI and PCIX
  - PCIe busses are actually point-to-point
  - Between 1 and 32 lanes, each of which is a differential pair.
  - 500MB/s per lane
  - Max of 16GB/s per card -- I don’t know of any 32 lane cards, but 16 is common.
PCI/e

- “Peripheral Component Interconnect”
- The fastest general-purpose expansion option
  - Graphics cards
  - Network cards
  - High-performance disk controllers (RAID)
  - Slow stuff works fine too.
- Current generation in PCI Express (PCIe)
IO Interfaces

- Protocol Layer: What commands are legal and when? What do they mean?
- Transport layer: How do you send a chunk of data? Negotiating access?
- Physical layer: How do you send a bit? What shape should connector be? Voltage level?

- The protocol layer is largely independent of the lower layers
  - RS232 over USB
  - “IP over everything and everything over IP”
  - USB hard drives use the SCSI command set
Intel’s Latest: Tylersburg Chipset

- Intel Core i7 Processor family
  - QPI 25.6 GB/s
  - DDR3 memory 8.5 Gb/s
  - DDR3 memory 8.5 Gb/s
  - DDR3 memory 8.5 Gb/s

**North bridge**
- X58
- IOH
- PCIe Express 2.0
  - Graphics Support for Multi-card configurations: 1x16, 2x16, 4x8 or other combination
- up to 36 lanes
- 2 GB/s DMI

**South bridge**
- ICH10
- ICH10R
- 12 Hi-Speed USB 2.0 Ports; Dual EHCI; USB
  - 480 Mb/s each
- 6 PCI Express x1
  - 500 Mb/s each x1
- Intel Integrated 10/100/1000 MAC
  - GLCI
  - LCI
- Intel Gigabit LAN Connect

**Other features**
- Intel High Definition Audio
- 6 Serial ATA Ports; eSATA; Port Disable
- Intel Matrix Storage Technology
- Intel Turbo Memory with User
- Optional
- LPC or SPI
- BIOS Support
- Intel Extreme Tuning Support
Hard Disks

• Hard disks are amazing pieces of engineering
  • Cheap
  • Reliable
  • Huge.
Disk Density

1 Tb/square inch
Hard drive Cost

- Yesterday at newegg.com: $0.008 GB ($0.000008/MB)
- Desktop, 1.5 TB
The Problem With Disk: It’s Sloooooowww

- on-chip cache
- off-chip cache
- main memory
- Disk

<table>
<thead>
<tr>
<th></th>
<th>Cost</th>
<th>Access time</th>
</tr>
</thead>
<tbody>
<tr>
<td>on-chip cache</td>
<td>0.000008 $/MB</td>
<td>10,000,000 ns</td>
</tr>
<tr>
<td>off-chip cache</td>
<td>0.07 $/MB</td>
<td>60 ns</td>
</tr>
<tr>
<td>main memory</td>
<td>2.5 $/MB</td>
<td>5 ns</td>
</tr>
<tr>
<td>Disk</td>
<td>0.000008 $/MB</td>
<td>&lt; 1 ns</td>
</tr>
</tbody>
</table>

The on-chip cache is the fastest with access time less than 1 ns, but it is also the most expensive at 2.5 $/MB. The main memory, used for temporary storage, is cheaper at 0.07 $/MB but has a longer access time of 60 ns. The disk is the slowest with an access time of 10,000,000 ns, but it is the cheapest at 0.000008 $/MB.
Why Are Disks Slow?

- They have moving parts :-(
  - The disk itself and the a head/arm
- The head can only read at one spot.
- High end disks spin at 15,000 RPM
  - Data is, on average, 1/2 an revolution away: 2ms
  - Power consumption limits spindle speed
  - Why not run it in a vacuum?
- The head has to position itself over the right “track”
  - Currently about 150,000 tracks per inch.
  - Positioning must be accurate with about 175nm
  - Takes 3-13ms
Making Disks Faster

- Caching
  - Everyone tries to cache disk accesses!
  - The OS
  - The disk controller
  - The disk itself.

- Access scheduling
  - Reordering accesses can reduce both rotational and seek latencies
Solid-state disks (SSDs)

- In your iPod, in your laptop, beginning to show up in data centers
- Use NAND flash memory instead of a spinning disk
NAND Flash

- Three operations
  - Erase a block (very slow)
  - Program a page (slower)
  - Read a page (fast)
- SLC – one bit per xtr
  - Fast, less dense
- MLC – two bits per xtr
  - Denser, slower, cheaper
- Reliability decreases with program/erase cycles

<table>
<thead>
<tr>
<th>Block 0</th>
<th>Block 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 0</td>
<td>Page 0</td>
</tr>
<tr>
<td>Page 1</td>
<td>Page 1</td>
</tr>
<tr>
<td>Page 2</td>
<td>Page 2</td>
</tr>
<tr>
<td>Page 3</td>
<td>Page 3</td>
</tr>
<tr>
<td>Page 4</td>
<td>Page 4</td>
</tr>
<tr>
<td>Page 5</td>
<td>Page 5</td>
</tr>
</tbody>
</table>
Flash’s Internal Structure

- Flash stores bit on a “floating gate” in a floating gate transistor.
- The gate is electrically isolated, so charge stays put.
- Charge can be pulled on and off the gate using large voltages on the terminals of the transistor.
- The charge on the gate affects the transistors switching characteristics, which allows us to read the bits out.
The Flash Chain

- Floating gate transistors are arranged in series as “chains”
- This allows for very high density: \(4F^2/\text{bit}\)
- DRAM is \(17F^2/\text{bit}\)
- Makes reading and writing slow -- all the other gates are in the way
Flash Blocks

- Many parallel chains form a block.
- A slice across the chains is a page.
- Read/Program operations affect one bit in each chain.
- Erases effect all the bits in a chain.
Individual Flash Chip Performance

- Transfer on and off the chip: 40MHz by 8 bits
  - Silly historical reasons. Currently a move is underway to
    133MHz by 16 bits
- DRAM is currently ~1GHz

- Operation latencies
  - 25-35us for reads
  - 200-2000us for programs
  - 1-3ms for erase
Reliability

• Flash wears out with use
  • Break down in the insulation around the floating gates lets charge leak off the gate.
• For MLC devices -- 10k program/erase cycles
• For SLC devices -- 100k program/erase cycles
• You can “burn a hole” in a flash chip in about 12 hours.
Wear Leveling

- SSDs must spread out program/erase operations evenly across the flash chips.
- They maintain a table that maps “Logical block addresses” (i.e., disk block addresses) to flash pages/blocks.
- This “Flash translation layer” reduces performance and adds complexity.
- SSD performance can be erratic.
- FTLs also provide error correction to recover from bit errors (which can be frequent, esp. for MLC).
- This is the key differentiator between SSDs.
SSDs vs HDD

- Expensive
  - SSD -$3/Gig (80GB Intel SSD)
  - Disk - $0.08/Gig (2TB seagate drive)

- Fast
  - IOPS
    - Random IO operations per second (IOPS)
    - SSD -- 3000/s for writes, 35,000 for reads (says Intel)
    - Disk -- 1/15ms = 66/s
  - BW MB/s
    - SSD -- 170MB/s write; 250MB/s read (max)
    - Disk -- 125MB/s (max)
  - Latency
    - SSD -- 75 microseconds for reads; intel won’t say for writes(!) probably 100s-1000s of microseconds
    - Disk -- 4-8ms

- Low power
  - SSD -- 2.4W max 0.06W idle
  - Disk -- 6.56W active; 5W idle
  - How often is your disk idle?
RAID!

• Redundant Array of Independent (Inexpensive) Disks
• If one disk is not fast enough, use many
  • Multiplicative increase in bandwidth
  • Multiplicative increase in Ops/Sec
  • Not much help for latency.
• If one disk is not reliable enough, use many.
  • Replicate data across the disks
  • If one of the disks dies, use the replica data to continue running and re-populate a new drive.
• Historical foot note: RAID was last invented by one of the textbook authors (Patterson)
RAID Levels

- There are several ways of ganging together a bunch of disks to form a RAID array. They are called “levels”
- Regardless of the RAID level, the array appears to the system as a sequence of disk blocks.
- The levels differ in how the logical blocks are arranged physically and how the replication occurs.
- Popular levels: 0, 1, 5, 10.
RAID 0

- Double the bandwidth.
- For an n-disk array, the n-th block lives on the n-th disk.
- Worse for reliability
  - If one of your drives dies, all your data is corrupt-- you have lost every nth block.
RAID 1

- Double the read bandwidth.
- Must write to both disks.
- For an n-disk array, the n-th block lives on all disks
- Better for reliability
  - If one of your drives dies, use the other.