Linux for Researchers

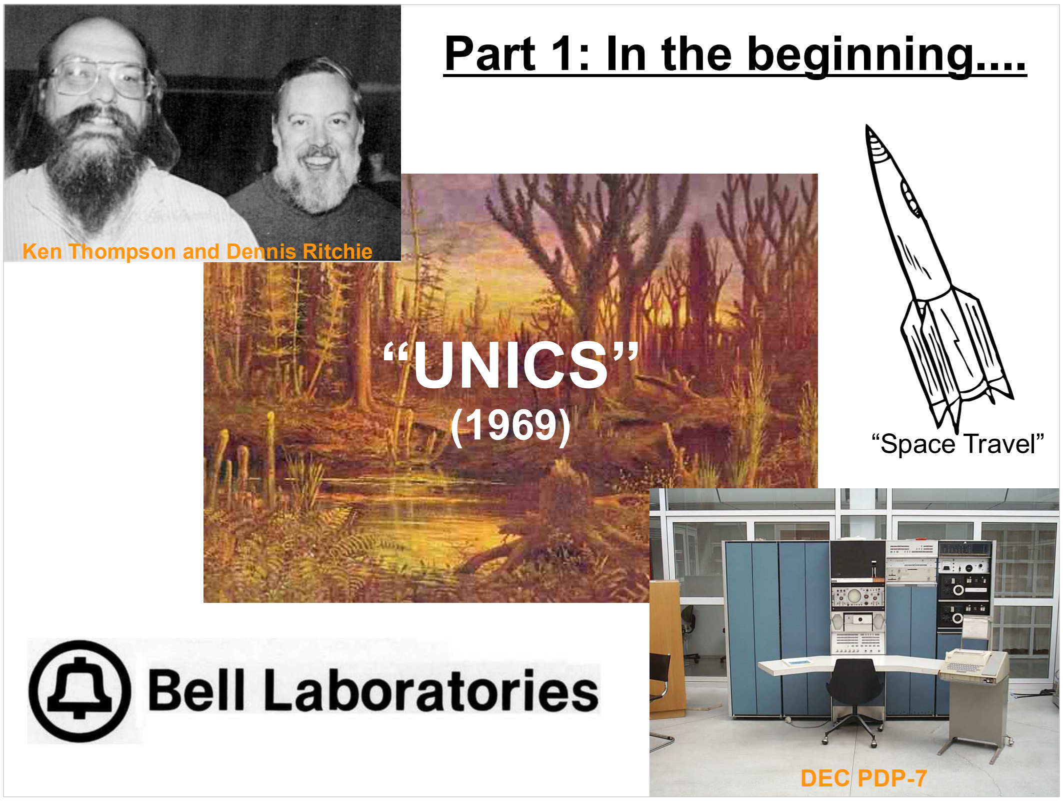
Chapter 1:
History and Hardware

In this talk I'd like to do two things:

1. Tell you a little about the history of Linux, so you have a better idea of what Linux is, and

2. I'll try to give you a basic familiarity with the hardware components of a modern PC.

This talk will be a little atypical, since we'll cover a lot of stuff very quickly, and much of it won't be specific to Linux.



Part 1: In the beginning....

Ken Thompson and Dennis Ritchie

“UNICS”

(1969)

“Space Travel”

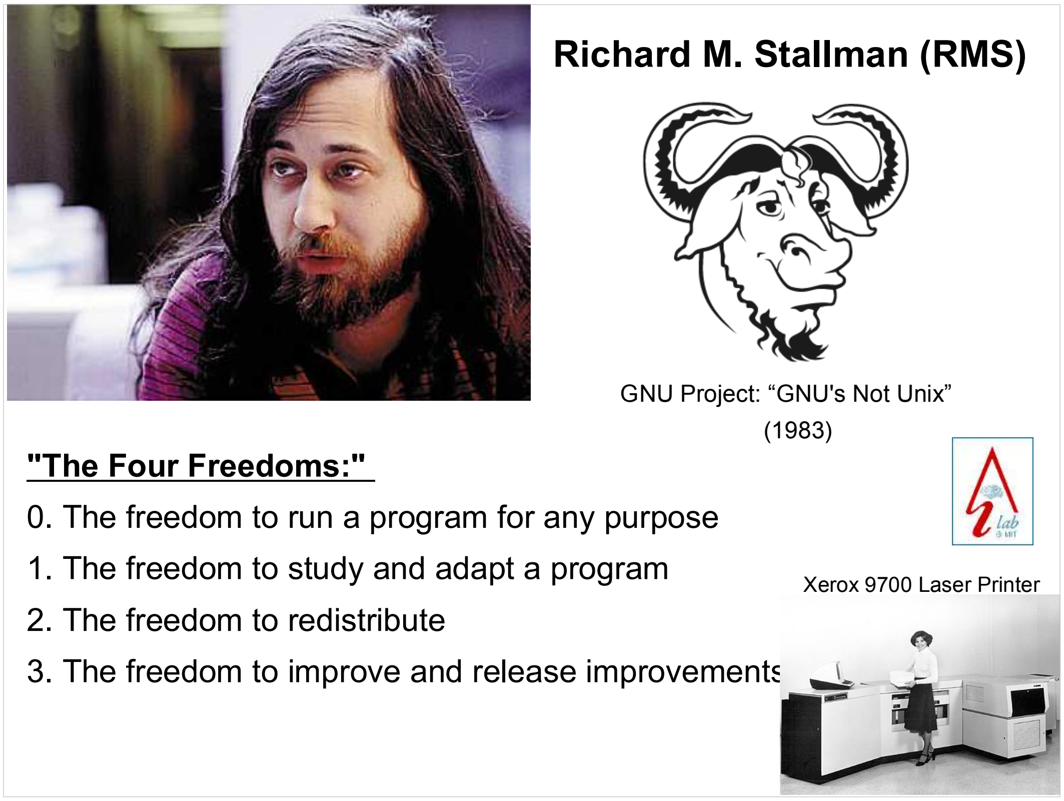
Bell Laboratories

DEC PDP-7

In 1969, something called “UNICS” crawled out of the primordial ooze. Ken Thompson and Dennis Ritchie of AT&T Bell Labs were working on a GE mainframe under an operating system called MULTICS.

Thompson had written a game called “space travel” that ran very slowly on the GE machine. The lab had recently acquired a DEC PDP-7 computer, and Thompson began re-writing his game in assembly code for the PDP-7, where he hoped it would run faster.

This effort eventually grew into an entire operating system called UNICS, in contrast to MULTICS. AT&T originally took no interest, but eventually saw the value of the new operating system and began selling it under its new name “UNIX”. It was pretty successful.... http://www.levenez.com/unix/



Richard M. Stallman (RMS)

GNU Project: “GNU's Not Unix”
(1983)

<u>"The Four Freedoms:"</u>

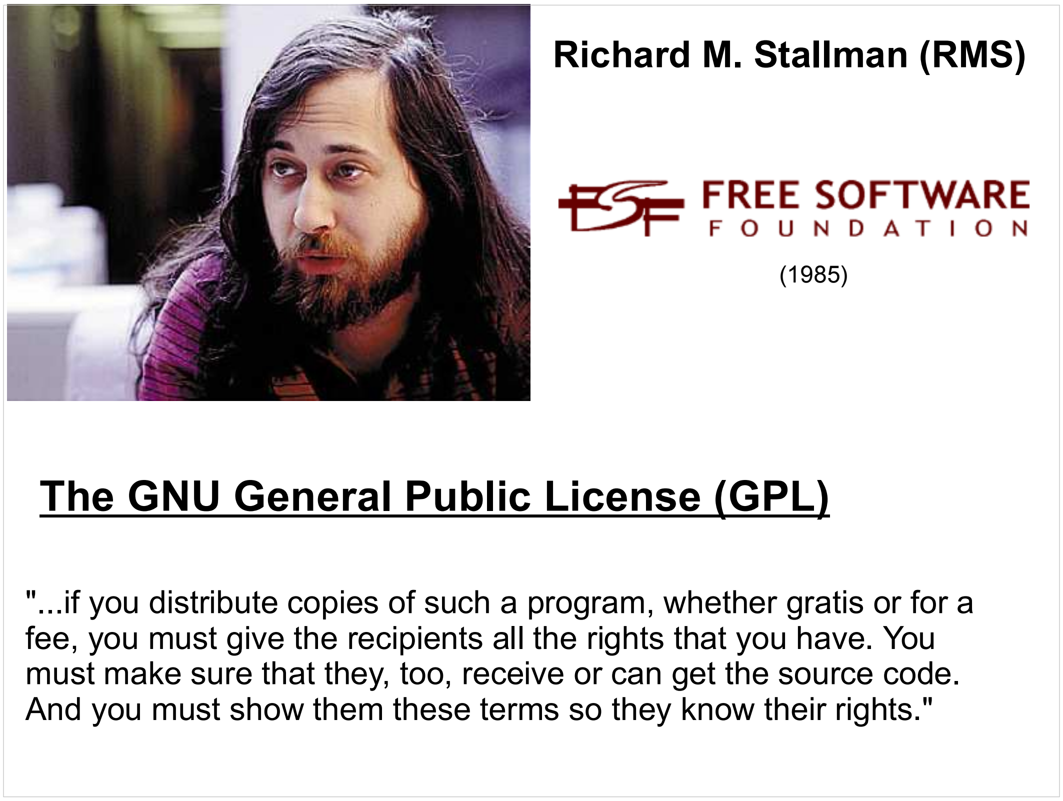
0. The freedom to run a program for any purpose
1. The freedom to study and adapt a program
2. The freedom to redistribute
3. The freedom to improve and release improvements

Xerox 9700 Laser Printer

A few years later we meet a very bright student named Richard M. Stallman (RMS). RMS graduated magna cum laude from Harvard and entered grad school (in physics) at MIT. As an undergrad and a grad, he worked in MIT's Artificial Intelligence Lab, and made significant contributions to the field of AI.

When he began his work, he was immersed in the hacker culture of the time, which valued the free exchange of ideas. Things were beginning to change, though. One example involved the first laser printer, made by Xerox. RMS had been able to get source code from Xerox for their earlier printers, so he could modify the printers' behavior. (Xerox was, after all, in the business of selling printers, not software). But Xerox refused to give him the code for their new laser printer, saying that it was proprietary.

As time went on, MIT spawned off many companies around proprietary software, and even began selling Stallman's own code. All of this upset RMS greatly. He believed that users and developers of software should have what he called “four freedoms”, shown above. Toward that end, in 1983, he began developing software for what he hoped would eventually be a complete, free (in terms of the “four freedoms”) operating system, which he called “GNU”, for “GNU's Not Unix” (the first of many recursive acronyms).



Richard M. Stallman (RMS)

FREE SOFTWARE FOUNDATION

(1985)

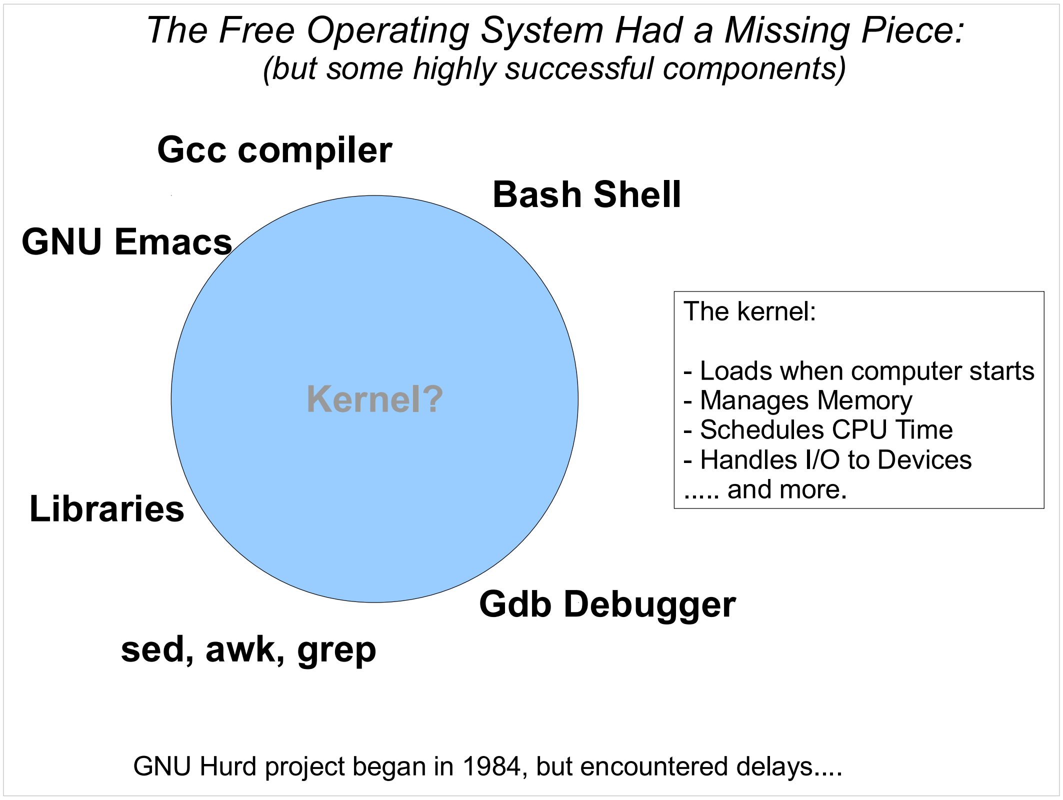
The GNU General Public License (GPL)

"...if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights."

To foster the development of free software, RMS founded the “Free Software Foundation” in 1985. His plan was that the FSF would hold copyrights on GNU software, and would license the use of the software under the terms of the GNU “General Public License”, which would ensure that the software always remained free.

http://www.fsf.org/

http://gplv3.fsf.org/



The Free Operating System Had a Missing Piece:
(but some highly successful components)

Gcc compiler

Bash Shell

GNU Emacs

Kernel?

The kernel:

- Loads when computer starts
- Manages Memory
- Schedules CPU Time
- Handles I/O to Devices
..... and more.

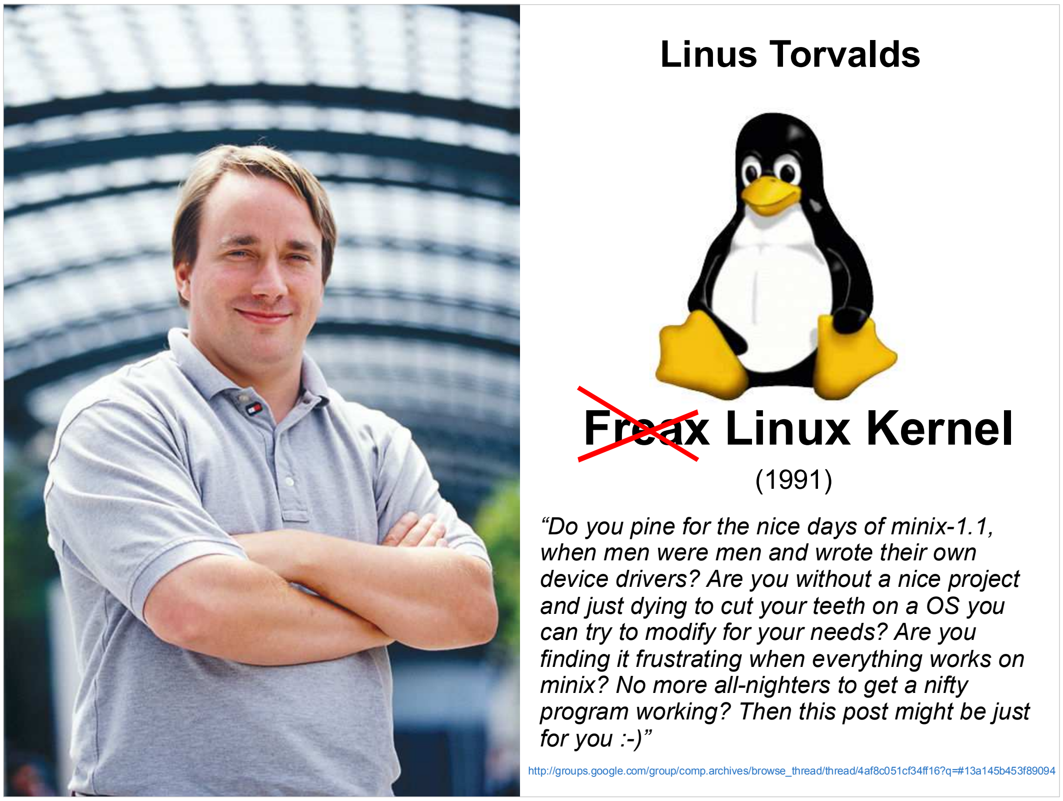
Libraries

Gdb Debugger

sed, awk, grep

GNU Hurd project began in 1984, but encountered delays....

By the late 80s, the GNU project had produced an impressive suite of tools, many of which were widely used (gcc and emacs, for example). These were largely independent of each other, and could be used on a wide range of common operating systems, replacing or supplementing tools already found in those operating systems. But a missing piece was still necessary to make GNU a complete operating system on it own. That piece was the kernel. In 1984 RMS had started the Hurd project, with the goal of making a GNU kernel, but it wasn't getting anywhere fast.



Linus Torvalds

~~Freax~~ Linux Kernel

(1991)

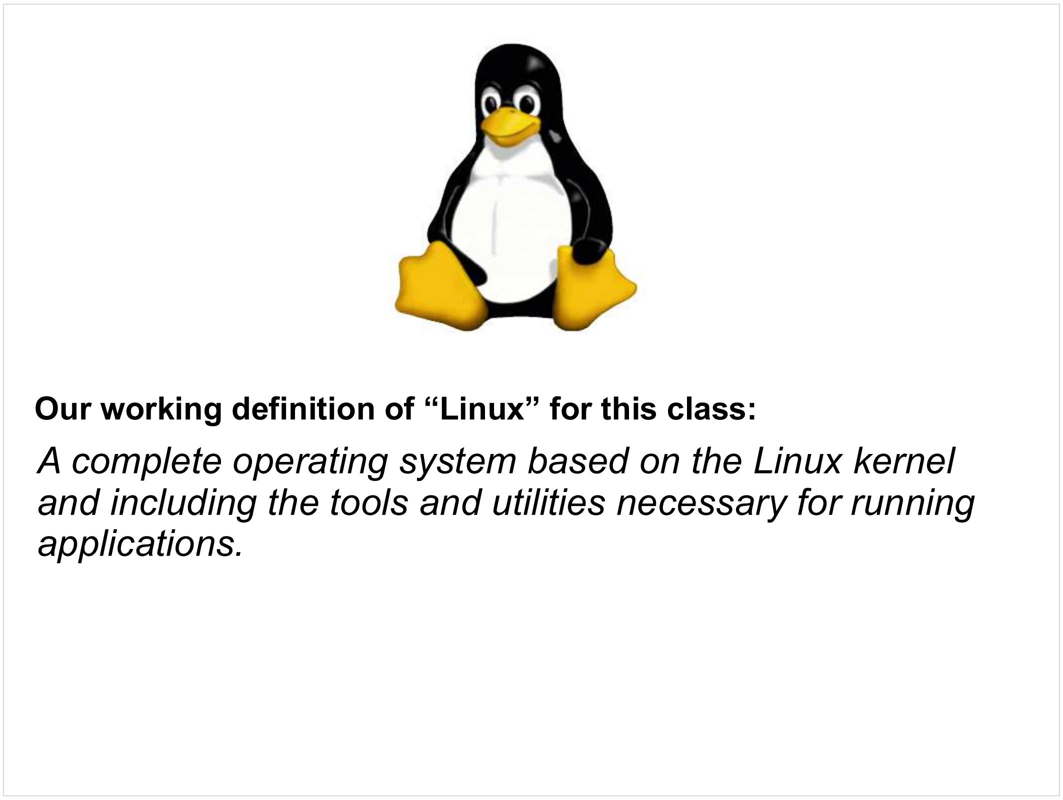
“Do you pine for the nice days of minix-1.1, when men were men and wrote their own device drivers? Are you without a nice project and just dying to cut your teeth on a OS you can try to modify for your needs? Are you finding it frustrating when everything works on minix? No more all-nighters to get a nifty program working? Then this post might be just for you :-)”

http://groups.google.com/group/comp.archives/browse_thread/thread/4af8c051cf34ff16?q=#13a145b453f89094

Then, in 1991, we meet another bright student. Linus Torvalds was a Finnish engineering student. He'd been inspired by an Operating Systems course he'd taken that used Andrew Tanenbaum's “Minix”, a toy implementation of Unix intended for teaching purposes.

Minix was simple enough to understand, and it ran on cheap, readily available PCs, but it wasn't free. Linus decided to try writing his own Unix-like kernel, to be released under the GPL, and in 1991 posted the message above to the comp.os.minix newsgroup, inviting other developers to download his code and help him work on it.

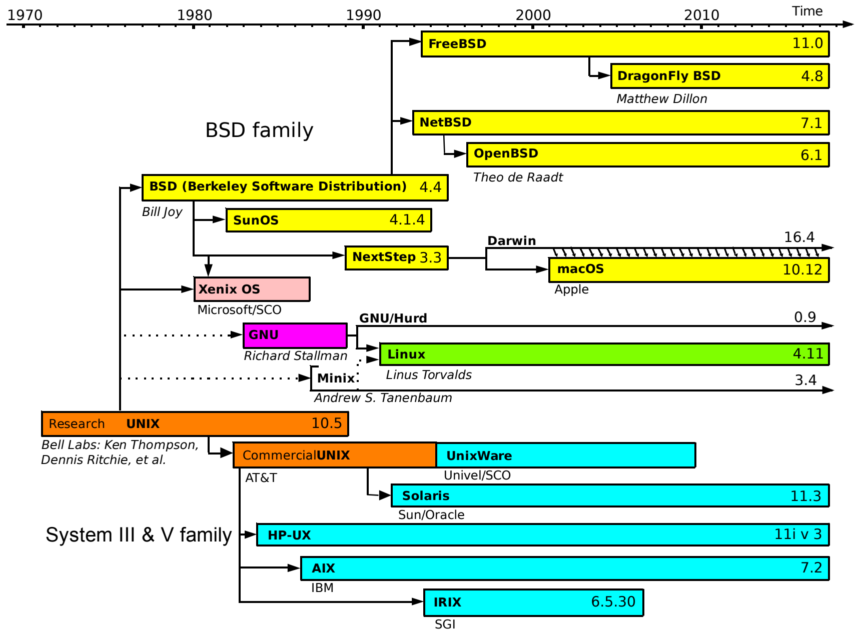
He originally called the kernel “FREAX”, for “Free Unix”, but the administrator of the download site named the directory “linux”, for “Linus's Unix”, and the name stuck.

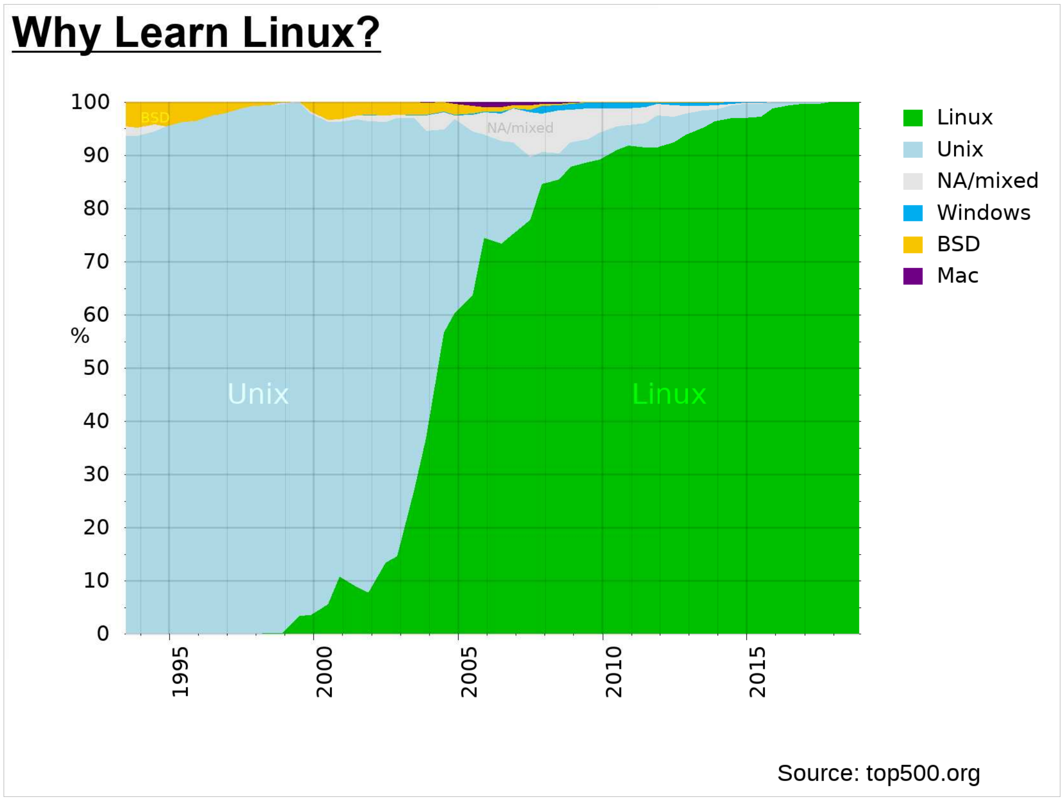


Our working definition of “Linux” for this class:

A complete operating system based on the Linux kernel and including the tools and utilities necessary for running applications.

Strictly speaking “Linux” just refers to the kernel that Linus Torvalds began developing. But when people say “Linux” these days they often mean a complete operating system that uses the Linux kernel. That's the definition we'll use for this class.


1970
1980
1990
2000
2010
Time
FreeBSD 11.0
DragonFly BSD 4.8
Matthew Dillon
NetBSD 7.1
OpenBSD 6.1
Theo de Raadt
BSD family
BSD (Berkeley Software Distribution) 4.4
Bill Joy
SunOS 4.1.4
NextStep 3.3
Darwin
16.4
macOS 10.12
Apple
Xenix OS
Microsoft/SCO
GNU
Richard Stallman
GNU/Hurd
0.9
Linux 4.11
Linus Torvalds
Minix
Andrew S. Tanenbaum
3.4
Research UNIX 10.5
Bell Labs: Ken Thompson, Dennis Ritchie, et al.
CommercialUNIX
AT&T
UnixWare
Univel/SCO
Solaris 11.3
Sun/Oracle
System III & V family
HP-UX 11i v 3
AIX 7.2
IBM
IRIX 6.5.30
SGI

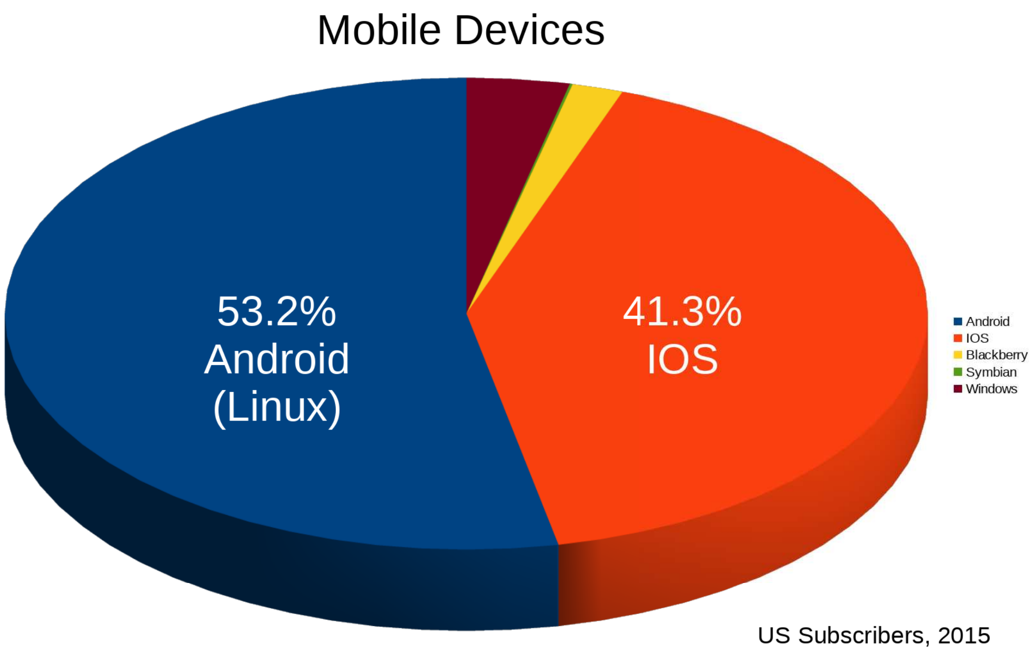


Why Learn Linux?

Source: top500.org

Linux-based computer systems are a mainstay in the world of scientific computing. In any laboratory setting where the research requires large amounts of data processing or computationally intensive calculations, you will routinely find Linux computers handling the workload. As of Nov 2010 92% (96%, counting Unix) of the world's top performing computer systems operate on Linux. Linux is the overwhelming choice for high-performance computing.

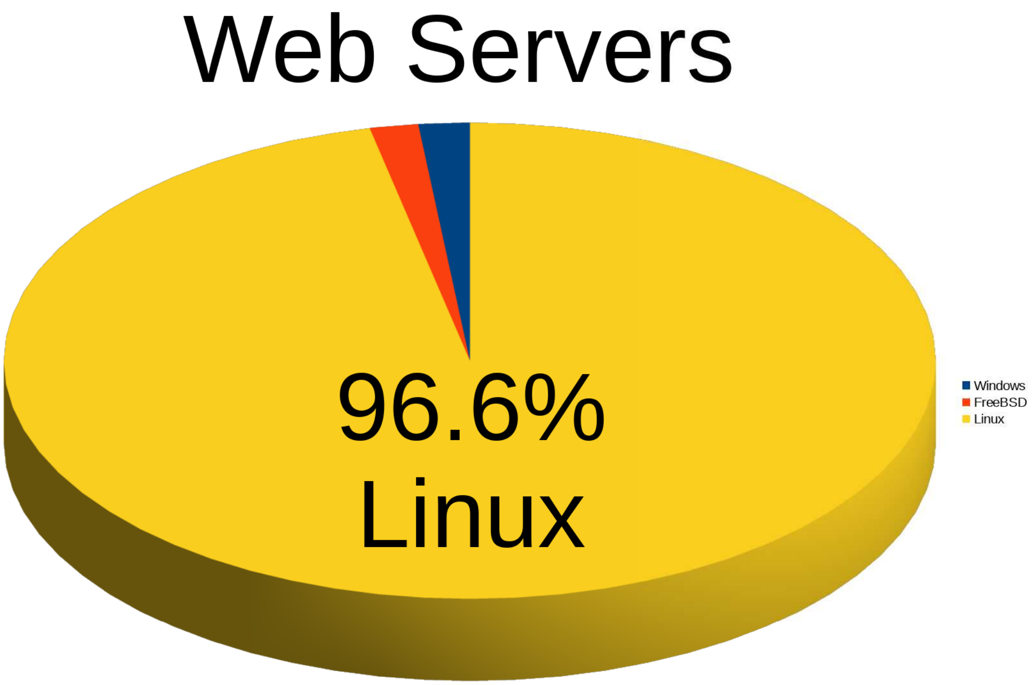
...and if you're not a scientist:



US Subscribers, 2015
Source: comscore.com

Mobile devices are a second area where Linux dominates the market. The Linux-based “Android” operating system ran 53% of mobile devices in 2015.

http://en.wikipedia.org/wiki/Usage_share_of_operating_systems



Web Servers

Top 1 million web servers by Alexa rank, 2015
Source: w3cook.com

On the server side, 97% of the top 1 million web servers run Linux.

AUTOMOTIVE GRADE LINUX
Ford
HONDA
The Power of Dreams
mazda
MITSUBISHI MOTORS
SUBARU
NISSAN
TOYOTA
SURROUND SOUND
3D
OFF
ON
STEREO
BASS
TREBLE
SUB
BALANCE / FADE
MAR 25
04:12 PM
45 SOUTH AVE
0 mph
0°C
TITLE
Smart Device Link
MAR 26
03:09 PM
82% ~215MI
SAN FRANCISCO
L FRONT TIRE
R FRONT TIRE
CHILD LOCK ACTIVATED
L REAR TIRE
R REAR TIRE
N
28.00 KW-H / MI
OFF

Part 2: PC Hardware

IBM PC 5150
(1981)

* Price competitive with Apple's home computers

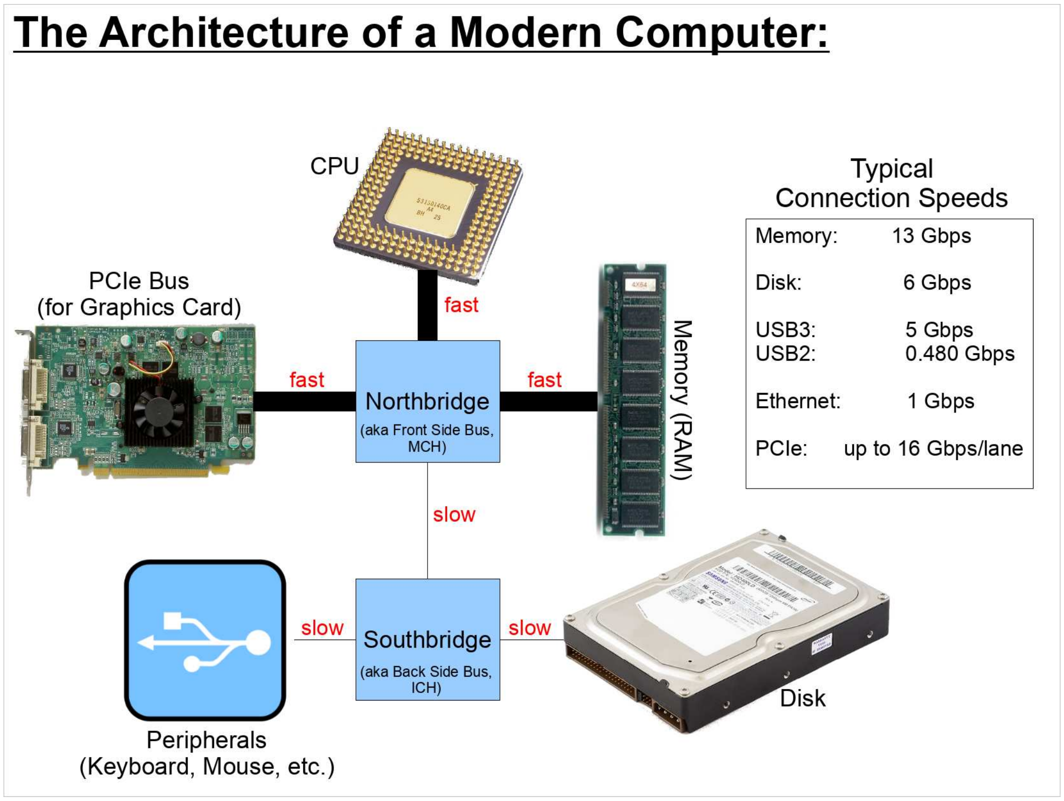
* Largely non-proprietary architecture could be cloned or re-engineered easily

* Ecosystem of clones created market for software vendors

Before 1981 people had “home computers”, but there was no such thing as a “PC”. Then, in 1981, IBM decided that there was money to be made in the burgeoning home computer market. They released their first home computer, called the IBM Personal Computer 5150.

Its price was competitive with the popular Apple home computers, and it was based largely on non-proprietary technology, which made it possible for other companies to clone the PC and sell PC-compatible computers. This created an enormous new market of compatible computers for hardware and software vendors.

Competition drove prices down, and consumers valued the variety of products available for the PC, and valued the IBM name. PCs came to dominate the low-priced computer market.



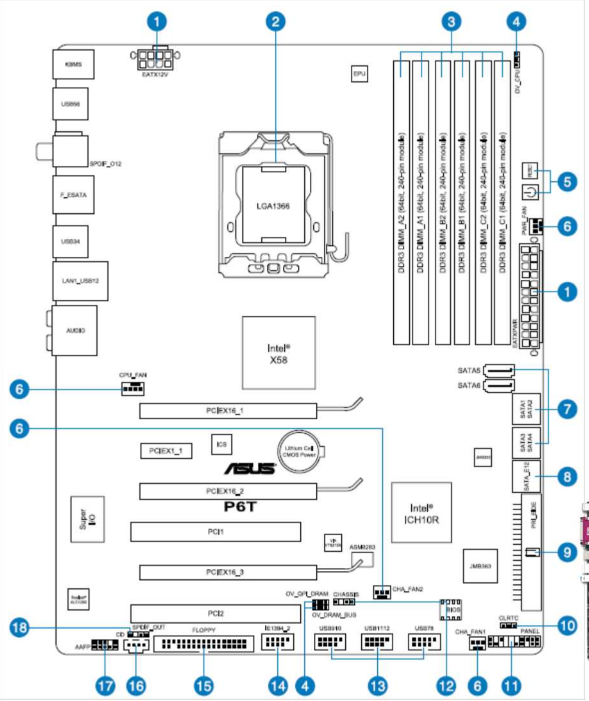
The Architecture of a Modern Computer:

Typical Connection Speeds

Memory:	13 Gbps
Disk:	6 Gbps
USB3:	5 Gbps
USB2:	0.480 Gbps
Ethernet:	1 Gbps
PCIe:	up to 16 Gbps/lane

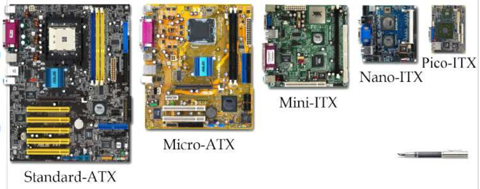
Today's PCs are still architecturally very similar to the original IBM PC. Some components to pay particular attention to are the CPU, which does the computer's thinking, and the Northbridge and Southbridge, which handle fast and slow I/O, respectively. Northbridge and Southbridge are often referred to collectively as the “chipset”. I've shown relative speeds for some of the connections for comparison. Note that disk access is much slower than memory access. We'll come back to this later.

A Typical Modern Motherboard with an “ATX” Form Factor:

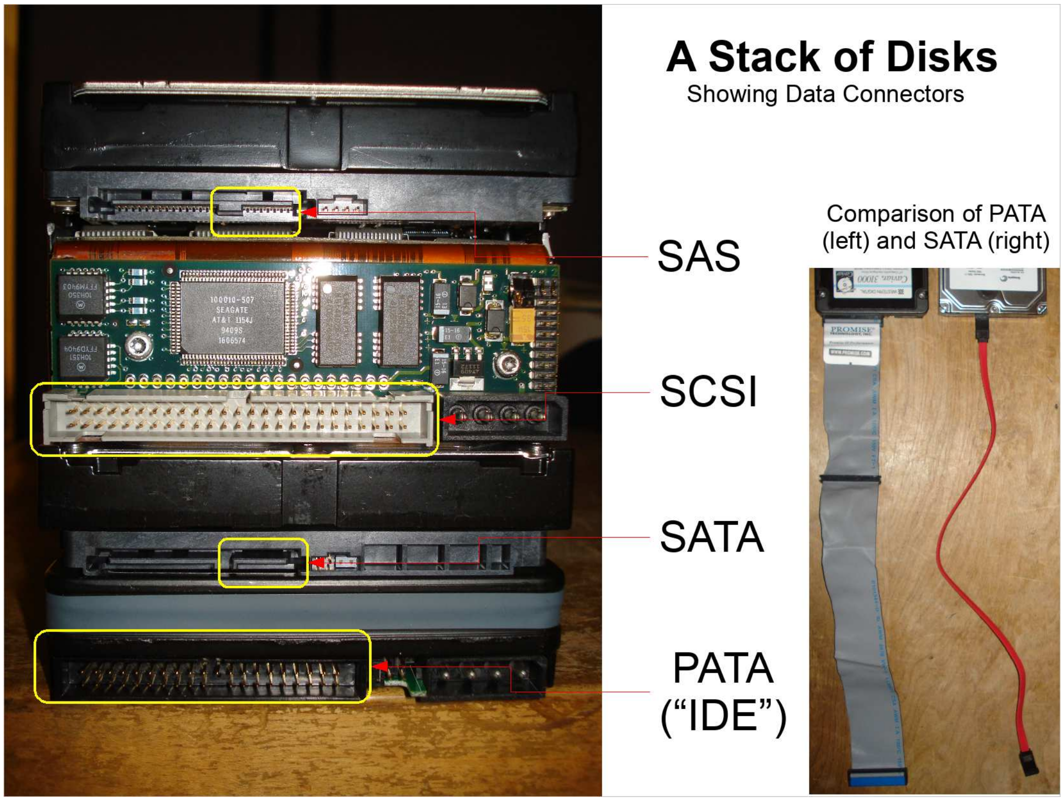


Connectors/Jumpers/Slots	
1.	ATX power connectors (24-pin EATXPWR, 8-pin EATX12V)
2.	LGA1366 CPU Socket
3.	DDR3 DIMM slots
4.	CPU / DRAM Bus / QPI DRAM overvoltage settings (3-pin OV_CPU; 3-pin OV_DRAM_BUS; 3-pin OV_QPI_DRAM)
5.	Onboard Power-on and Reset switch
6.	CPU, chassis, and power fan connectors (4-pin CPU_FAN, 3-pin CHA_FAN1-2, 3-pin PWR_FAN)
7.	ICH10R Serial ATA connectors [red] (7-pin SATA1-6)
8.	JMicron JMB322 Serial ATA connectors [orange] (7-pin SATA_E1, SATA_E2)
9.	IDE connector (40-1 pin PRI_EIDE)
10.	Clear RTC RAM (3-pin CLRTC)
11.	System panel connector (20-8 pin PANEL)
12.	Chassis intrusion connector (4-1 pin CHASSIS)
13.	USB connectors (10-1 pin USB78, USB910, USB1112)
14.	IEEE 1394a port connector (10-1 pin IE1394_2)
15.	Floppy disk drive connector (34-1 pin FLOPPY)
16.	Optical drive audio connector (4-pin CD)
17.	Front panel audio connector (10-1 pin AAFP)
18.	Digital audio connector (4-1 pin SPDIF_OUT)

Other Form Factors:



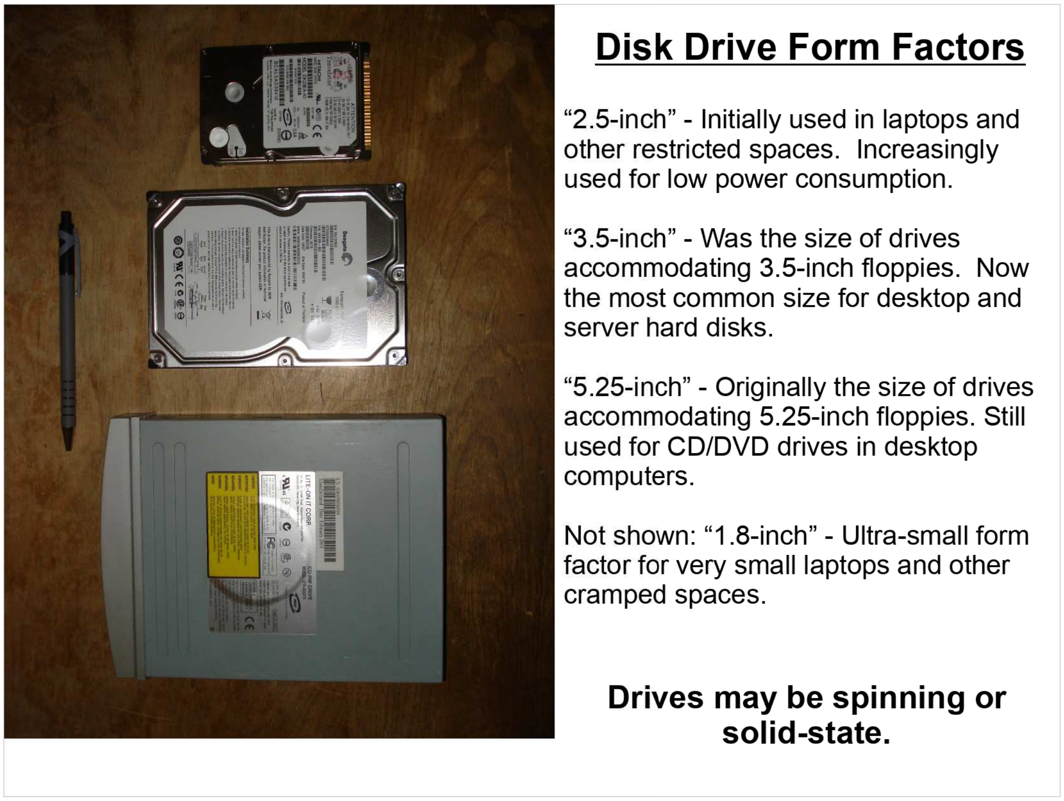
...and here's a picture of a motherboard's physical layout. Note that this is a motherboard with an ATX “form factor”. The form factor is just the size and shape of the motherboard. If your computer has an ATX motherboard, you know that you can replace it with another ATX motherboard from any manufacturer and it will fit into the same spot. The development of standards like this helped commoditize the PC. Owners and PC manufacturers can shop around for the best deals for parts.



In my experience, the two things most likely to fail in a computer are the disk and the power supply. Power supplies are trivial to replace, but disks contain your data, so I'm going to spend a little time talking about them.

Here are four different types of disks, all of the same width. Each is a standard "3.5-inch" disk, and would fit into the same slot as any of the others. The two most common types of disks are Parallel ATA (PATA) disks (sometimes loosely called "IDE" disks) and Serial ATA (SATA) disks. SATA is the successor to PATA, and is found almost universally in new computers.

The move from a parallel bus to a serial bus was driven by speed. With a parallel bus, crosstalk between adjacent wires becomes more and more of a problem with increasing signal frequency. Similarly, older SCSI disks (another parallel interface, found mostly in servers) are being phased out in favor of Serial-Attached SCSI (SAS).



Disk Drive Form Factors

"2.5-inch" - Initially used in laptops and other restricted spaces. Increasingly used for low power consumption.

"3.5-inch" - Was the size of drives accommodating 3.5-inch floppies. Now the most common size for desktop and server hard disks.

"5.25-inch" - Originally the size of drives accommodating 5.25-inch floppies. Still used for CD/DVD drives in desktop computers.

Not shown: "1.8-inch" - Ultra-small form factor for very small laptops and other cramped spaces.

Drives may be spinning or solid-state.

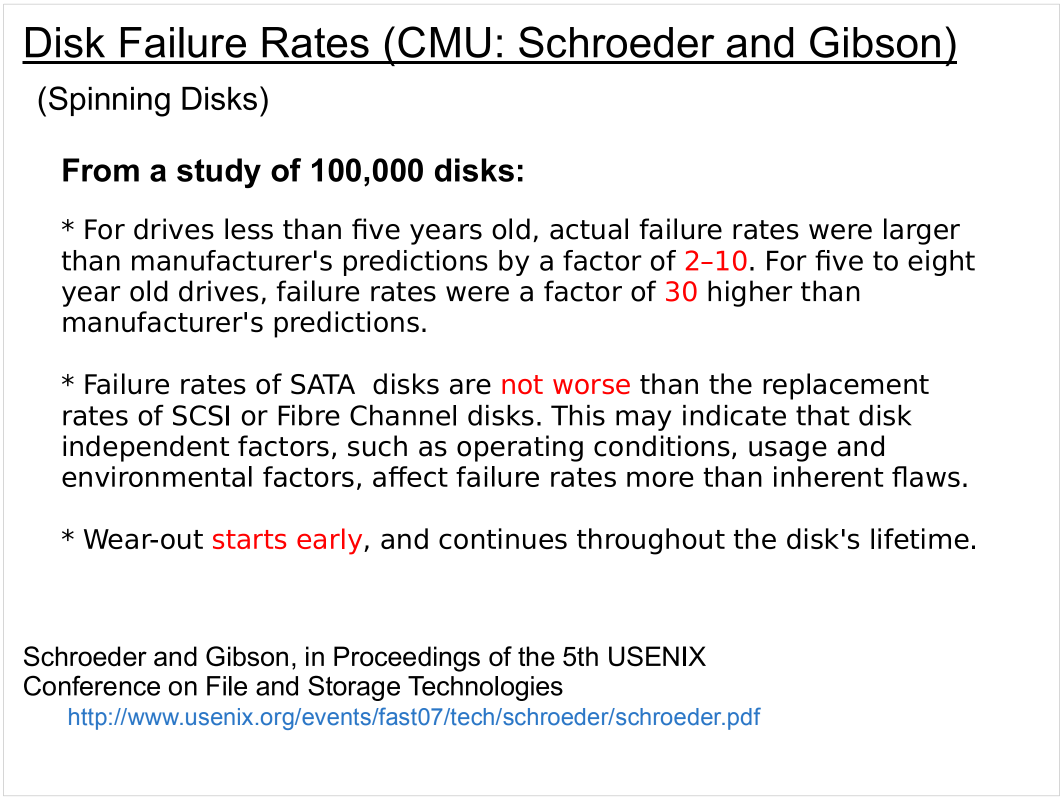
This shows some of the form factors used for disks. Note that the names of the form factors don't reflect the actual physical dimensions of the disks.

M.2 NVMe Drives:

M.2: Specifies physical dimentions and connector.

NVMe (“Non-Volatile Memory Express”): Protocol for talking to storage devices over PCIe bus.

Solid-state drives that communicate with the rest of the system through the PCIe bus, which is faster than traditional disk connections.



Disk Failure Rates (CMU: Schroeder and Gibson)

(Spinning Disks)

From a study of 100,000 disks:

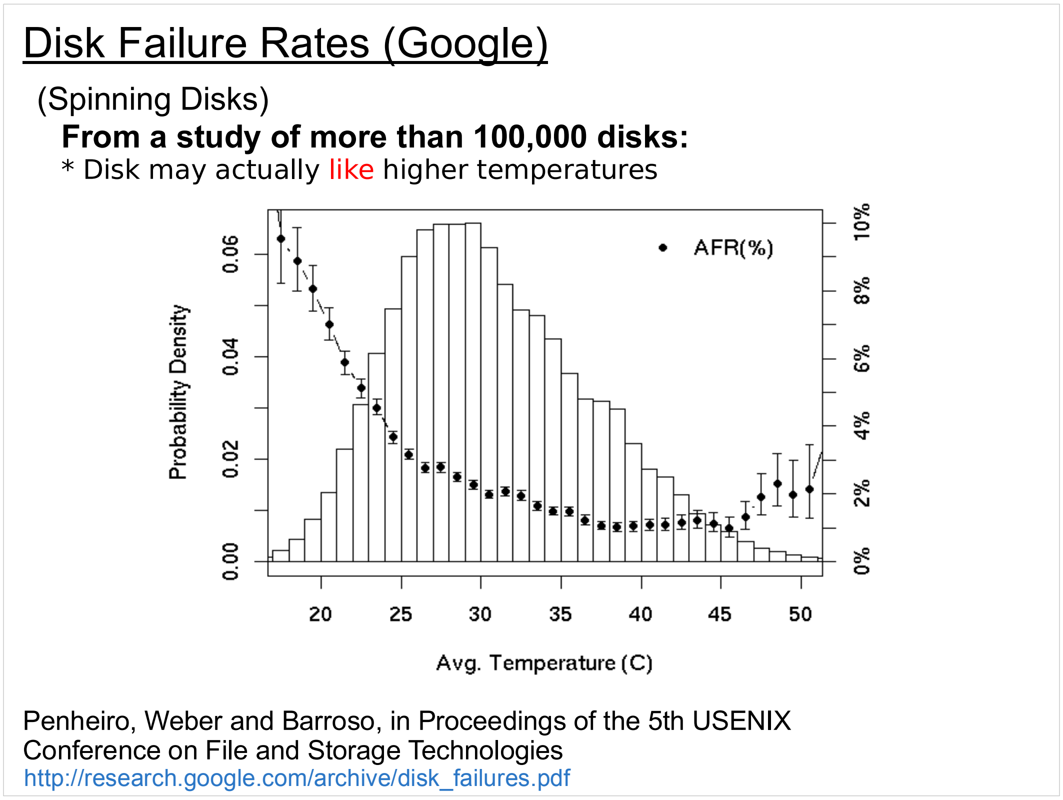
* For drives less than five years old, actual failure rates were larger than manufacturer's predictions by a factor of 2–10. For five to eight year old drives, failure rates were a factor of 30 higher than manufacturer's predictions.

* Failure rates of SATA disks are not worse than the replacement rates of SCSI or Fibre Channel disks. This may indicate that disk independent factors, such as operating conditions, usage and environmental factors, affect failure rates more than inherent flaws.

* Wear-out starts early, and continues throughout the disk's lifetime.

Schroeder and Gibson, in Proceedings of the 5th USENIX Conference on File and Storage Technologies
http://www.usenix.org/events/fast07/tech/schroeder/schroeder.pdf

There have been several recent studies of disk failure rates. I'll talk about a couple of particularly interesting ones, done at CMU and Google. These are some things to keep in mind when buying disks, thinking about backup strategies, or budgeting for replacement costs.



Disk Failure Rates (Google)

(Spinning Disks)

From a study of more than 100,000 disks:

* Disk may actually like higher temperatures

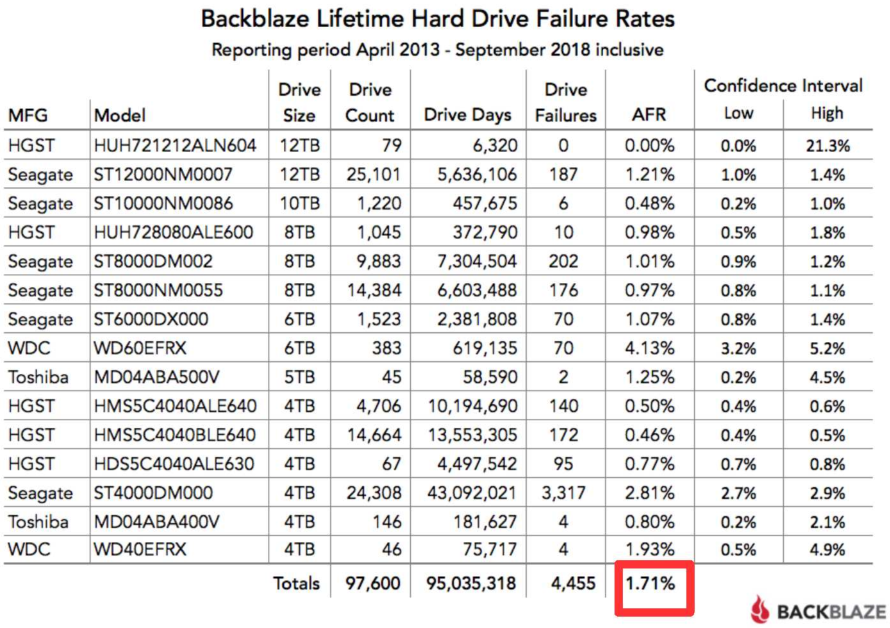
Penheiro, Weber and Barroso, in Proceedings of the 5th USENIX Conference on File and Storage Technologies
http://research.google.com/archive/disk_failures.pdf

The Google report confirms many of the CMU findings, and adds some interesting new finding. For example, our long-standing assumption that disks are more likely to fail at higher temperatures may not be correct. Maybe we could save money by leaving our server rooms at a higher temperature, or by eliminating some of the fans inside computers.

When manufacturers test disks, they can't run them for five years to see if they fail. Typically, they try to simulate long lifetimes by running the disks for a shorter time under extreme conditions (high temperature, for example). It may be that, because of this, manufacturers have been inadvertently selecting disk designs that prefer to run at higher temperatures.

Disk Failure Rates (BackBlaze)

(Spinning Disks)



Backblaze Lifetime Hard Drive Failure Rates

Reporting period April 2013 - September 2018 inclusive

		Drive	Drive		Drive		Confidence Interval	
MFG	Model	Size	Count	Drive Days	Failures	AFR	Low	High
HGST	HUH721212ALN604	12TB	79	6,320	0	0.00%	0.0%	21.3%
Seagate	ST12000NM0007	12TB	25,101	5,636,106	187	1.21%	1.0%	1.4%
Seagate	ST10000NM0086	10TB	1,220	457,675	6	0.48%	0.2%	1.0%
HGST	HUH728080ALE600	8TB	1,045	372,790	10	0.98%	0.5%	1.8%
Seagate	ST8000DM002	8TB	9,883	7,304,504	202	1.01%	0.9%	1.2%
Seagate	ST8000NM0055	8TB	14,384	6,603,488	176	0.97%	0.8%	1.1%
Seagate	ST6000DX000	6TB	1,523	2,381,808	70	1.07%	0.8%	1.4%
WDC	WD60EFRX	6TB	383	619,135	70	4.13%	3.2%	5.2%
Toshiba	MD04ABA500V	5TB	45	58,590	2	1.25%	0.2%	4.5%
HGST	HMS5C4040ALE640	4TB	4,706	10,194,690	140	0.50%	0.4%	0.6%
HGST	HMS5C4040BLE640	4TB	14,664	13,553,305	172	0.46%	0.4%	0.5%
HGST	HDS5C4040ALE630	4TB	67	4,497,542	95	0.77%	0.7%	0.8%
Seagate	ST4000DM000	4TB	24,308	43,092,021	3,317	2.81%	2.7%	2.9%
Toshiba	MD04ABA400V	4TB	146	181,627	4	0.80%	0.2%	2.1%
WDC	WD40EFRX	4TB	46	75,717	4	1.93%	0.5%	4.9%
		Totals	97,600	95,035,318	4,455	1.71%		

BACKBLAZE

https://www.backblaze.com/blog/2018-hard-drive-failure-rates/

Disk Failure Rates (Google: Schroeder et al.)

(Solid-State Disks)

From a study of millions of disk-hours[1]:

* Expensive SSDs are no more reliable than cheaper ones.

* Read error rates do not depend on load, but do depend on age.

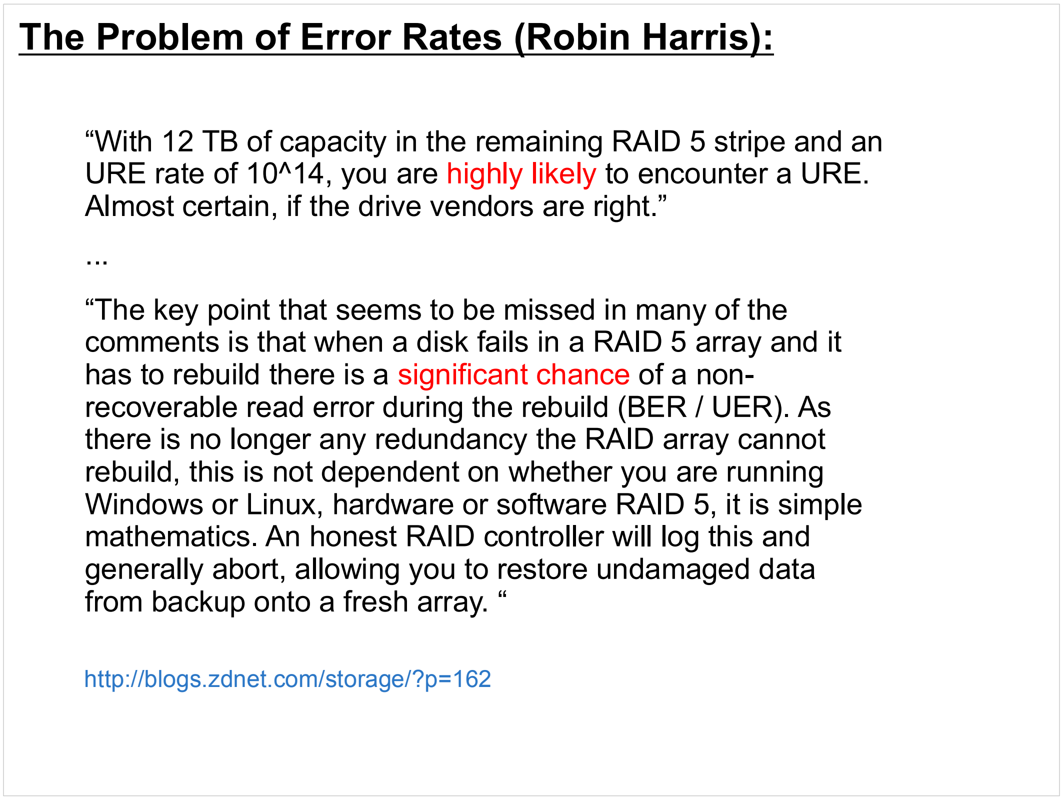
* Industry-standard error rate metrics are not meaningful.

* SSDs have a significantly higher rate of serious errors than do spinning disks.

[1] The exact number of disks is considered confidential, and isn't disclosed in the paper.

Schroeder, Lagisetty and Merchant, in Proceedings of the
14th USENIX Conference on
File and Storage Technologies (FAST '16)

https://www.usenix.org/conference/fast16/technical-sessions/presentation/schroeder



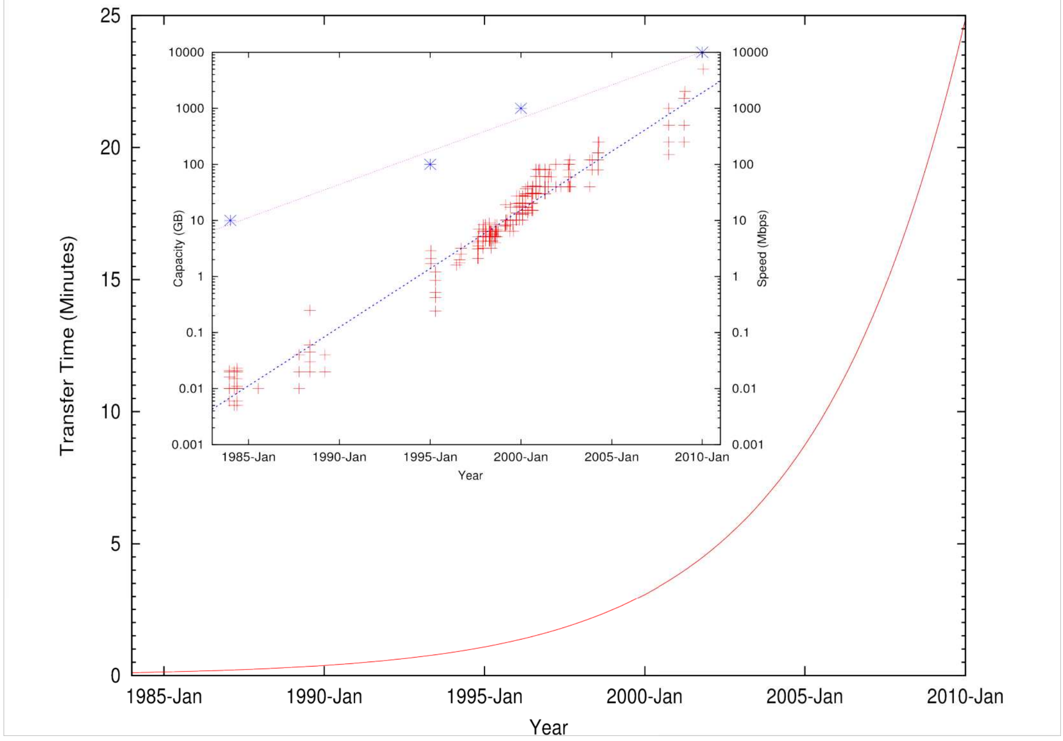
The Problem of Error Rates (Robin Harris):

> “With 12 TB of capacity in the remaining RAID 5 stripe and an URE rate of 10^14, you are highly likely to encounter a URE. Almost certain, if the drive vendors are right.”
>
> ...
>
> “The key point that seems to be missed in many of the comments is that when a disk fails in a RAID 5 array and it has to rebuild there is a significant chance of a non-recoverable read error during the rebuild (BER / UER). As there is no longer any redundancy the RAID array cannot rebuild, this is not dependent on whether you are running Windows or Linux, hardware or software RAID 5, it is simple mathematics. An honest RAID controller will log this and generally abort, allowing you to restore undamaged data from backup onto a fresh array. “

http://blogs.zdnet.com/storage/?p=162

This recent blog post by Robin Harris got a lot of attention. Manufacturers cite what's called an “Unrecoverable Read Error” (URE) rate for disks. This is a measure of the probability that a given bit of data will suddenly, and permanently, become unreadable. In the past, a URE rate of 1 in 10^14 has been acceptable, but as disks get bigger, it's becoming more and more likely that you'll encounter a URE when you read from the disk. The thing to note is that URE rates haven't kept up with disk sizes, and this is becoming a problem.

Disk Size vs. Network Speed:



Here's something I noticed recently. The smaller, inset plot shows how disk capacity (red marks) and ethernet speed (blue marks) have increased over the years. Note that disk capacity doubles approximately every two years, but ethernet speed only doubles every four years or so. To see what this means, look at the larger graph.

This graph shows the amount of time needed to transfer the entire contents of a current disk to another similar disk across the network, assuming that the only limiting factor is network speed. As you can see, this transfer time is steadily increasing, because network speeds aren't keeping pace with the increasing capacity of disks.

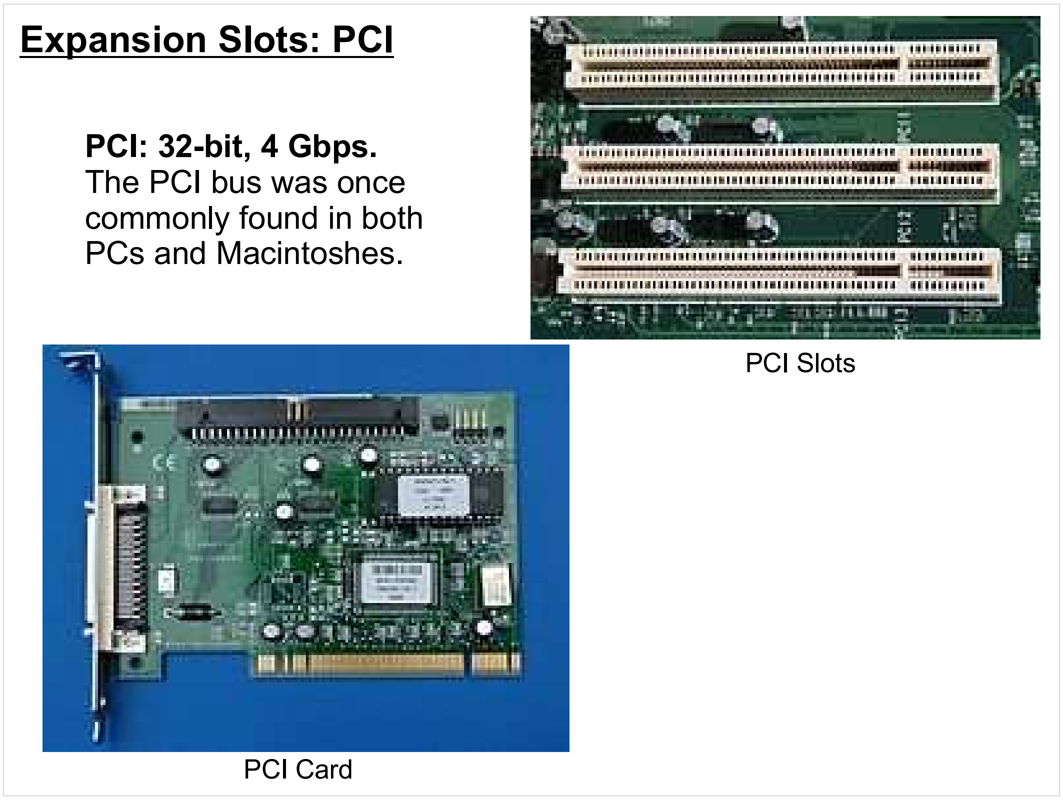
If this trend continues, it means that, in the future, we'll need to think more and more carefully about where our data lives.

Growth of Disk Capacity:

Mark Kryder, Seagate

"Since the introduction of the disk drive in 1956, the density of information it can record has swelled from a paltry 2,000 bits to 100 billion bits (gigabits), all crowded in the small space of a square inch. That represents a 50-million-fold increase."

Chip Walter, "Kryder's Law", Scientific American, August 1, 2005



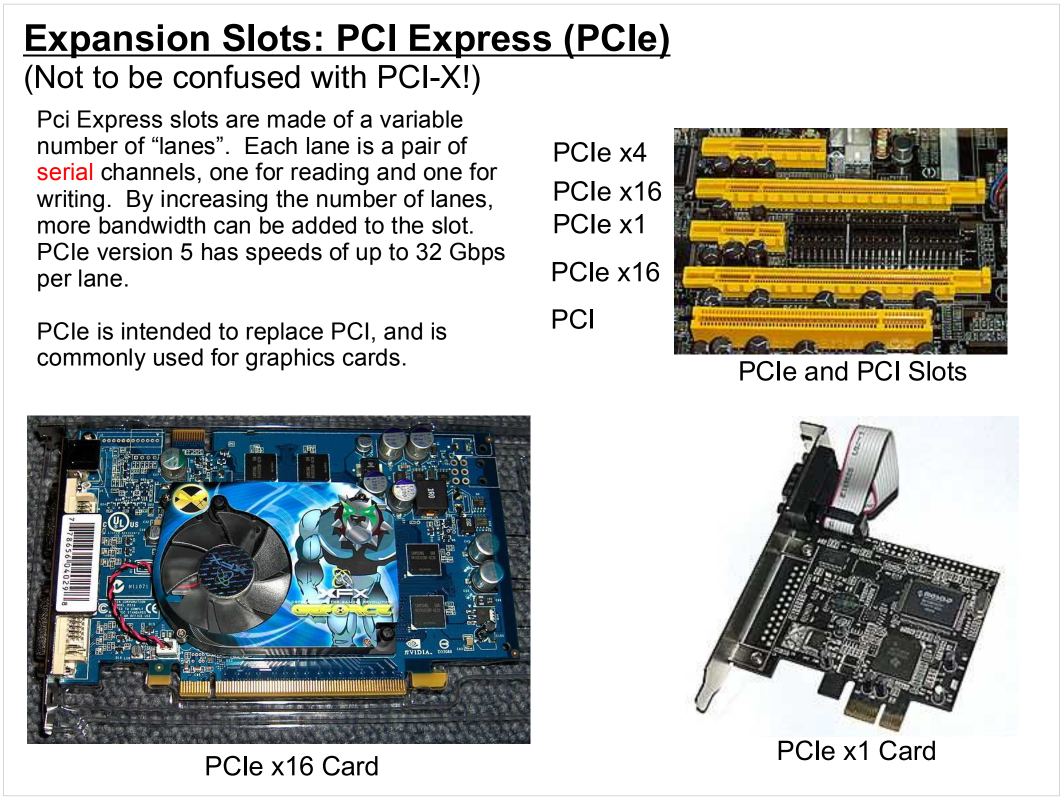
Expansion Slots: PCI

PCI: 32-bit, 4 Gbps.
The PCI bus was once commonly found in both PCs and Macintoshes.

PCI Slots

PCI Card

Now we'll start looking at some of the expansion slots used in PCs. The most common type of slot is still the PCI slot, but it has a couple of successors. One of them is PCI-x.



Expansion Slots: PCI Express (PCIe)

(Not to be confused with PCI-X!)

Pci Express slots are made of a variable number of “lanes”. Each lane is a pair of serial channels, one for reading and one for writing. By increasing the number of lanes, more bandwidth can be added to the slot. PCIe version 5 has speeds of up to 32 Gbps per lane.

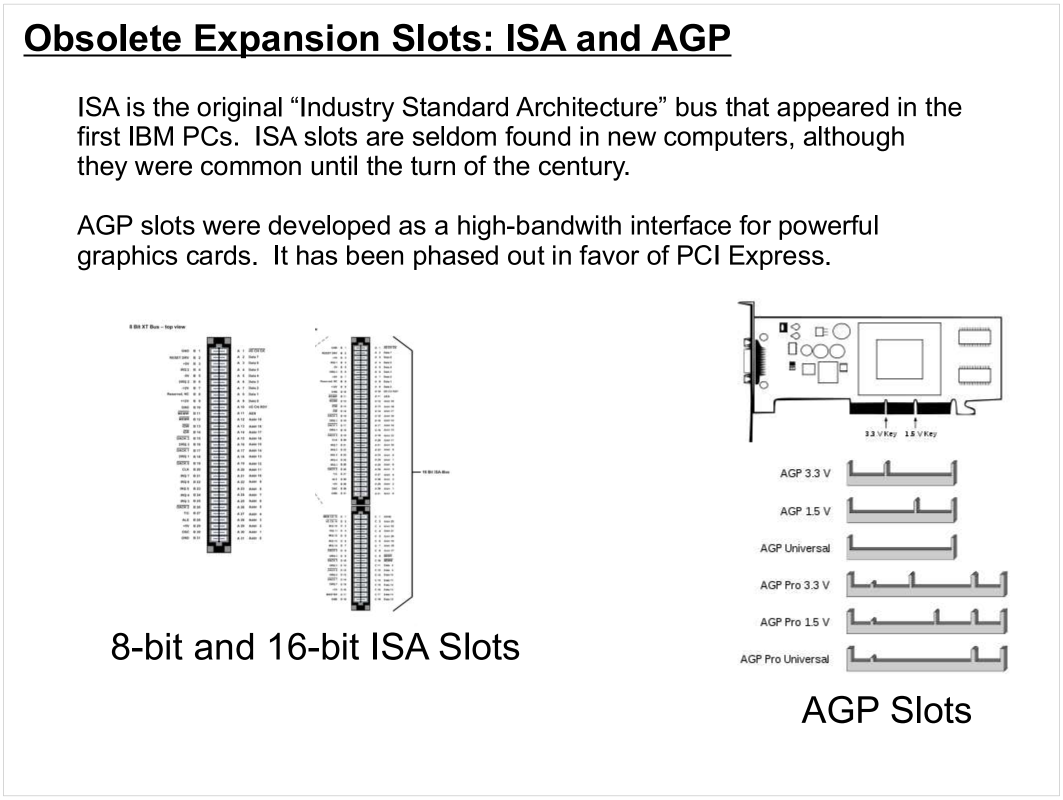
PCIe is intended to replace PCI, and is commonly used for graphics cards.

PCIe and PCI Slots

PCIe x16 Card

PCIe x1 Card

...and the other is PCI Express (PCIe). Confusing, no? Note that PCIe isn't backward-compatible with PCI, but it's becoming more and more common, and may eventually be the single standard expansion slot.



Obsolete Expansion Slots: ISA and AGP

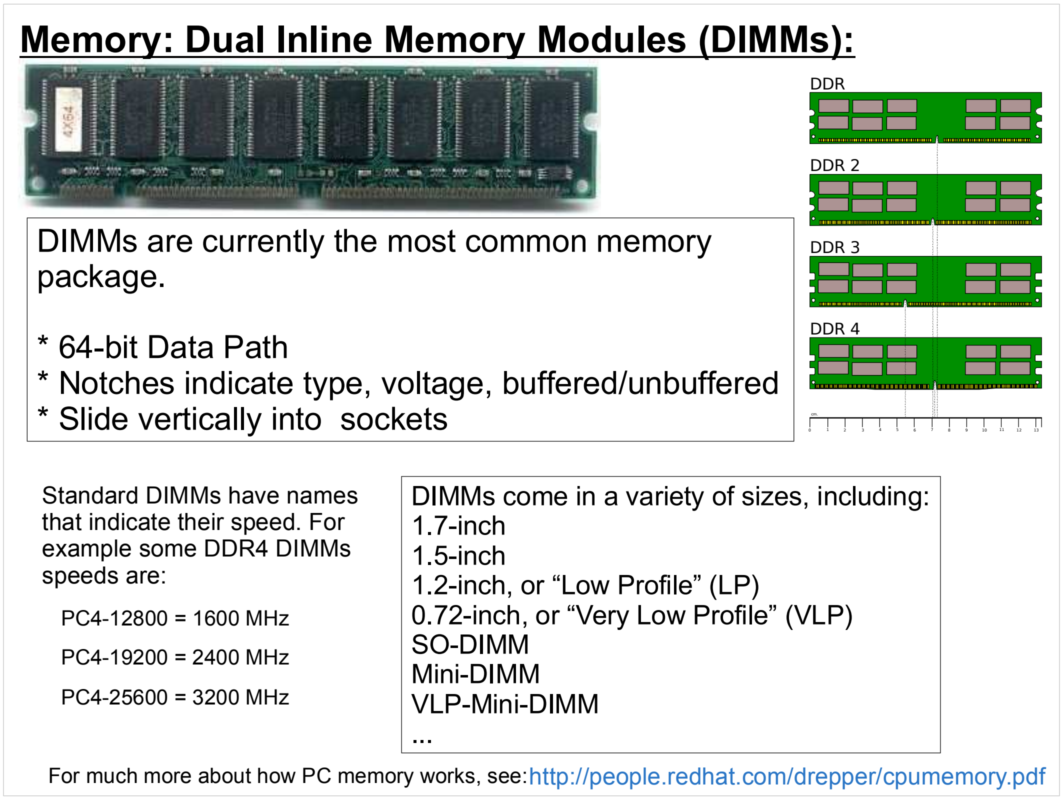
ISA is the original “Industry Standard Architecture” bus that appeared in the first IBM PCs. ISA slots are seldom found in new computers, although they were common until the turn of the century.

AGP slots were developed as a high-bandwith interface for powerful graphics cards. It has been phased out in favor of PCI Express.

8-bit and 16-bit ISA Slots

AGP Slots

Here are a couple of old types of expansion slot that aren't seen very much any more.



DDR
DDR 2
DDR 3
DDR 4

Memory: Dual Inline Memory Modules (DIMMs):

DIMMs are currently the most common memory package.

* 64-bit Data Path
* Notches indicate type, voltage, buffered/unbuffered
* Slide vertically into sockets

Standard DIMMs have names that indicate their speed. For example some DDR4 DIMMs speeds are:

PC4-12800 = 1600 MHz

PC4-19200 = 2400 MHz

PC4-25600 = 3200 MHz

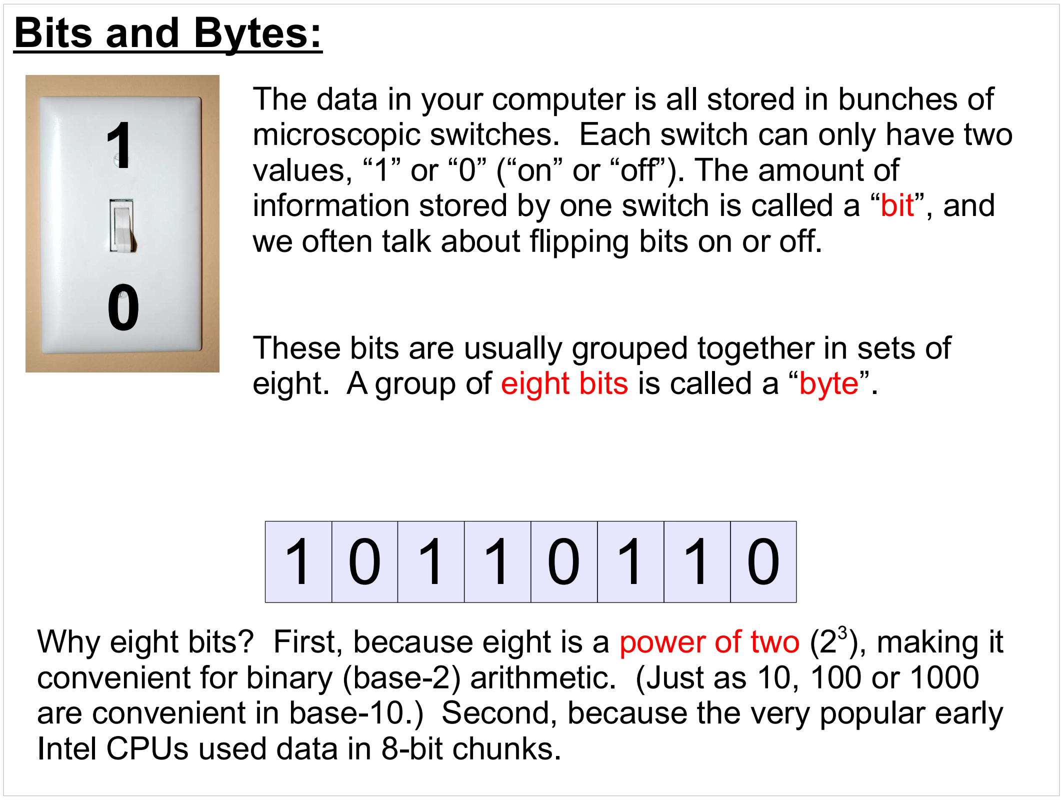
DIMMs come in a variety of sizes, including:
1.7-inch
1.5-inch
1.2-inch, or “Low Profile” (LP)
0.72-inch, or “Very Low Profile” (VLP)
SO-DIMM
Mini-DIMM
VLP-Mini-DIMM
...

For much more about how PC memory works, see:http://people.redhat.com/drepper/cpumemory.pdf

Again, note the standard form factors and speed designations. This means that when you buy memory for your computer you can shop around for, say, 1.5-inch PC3200 DIMMs.

The predecessor to DIMMs were SIMMs (Single Inline Memory Modules). These are seldom seen these days.

The computer's memory is a place to temporarily store data. Memory is “volatile”: data stored in memory disappears when the computer is wiped out. Compare this with data stored on a hard disk, which is saved until you explicitly delete it.



Bits and Bytes:

The data in your computer is all stored in bunches of microscopic switches. Each switch can only have two values, “1” or “0” (“on” or “off”). The amount of information stored by one switch is called a “bit”, and we often talk about flipping bits on or off.

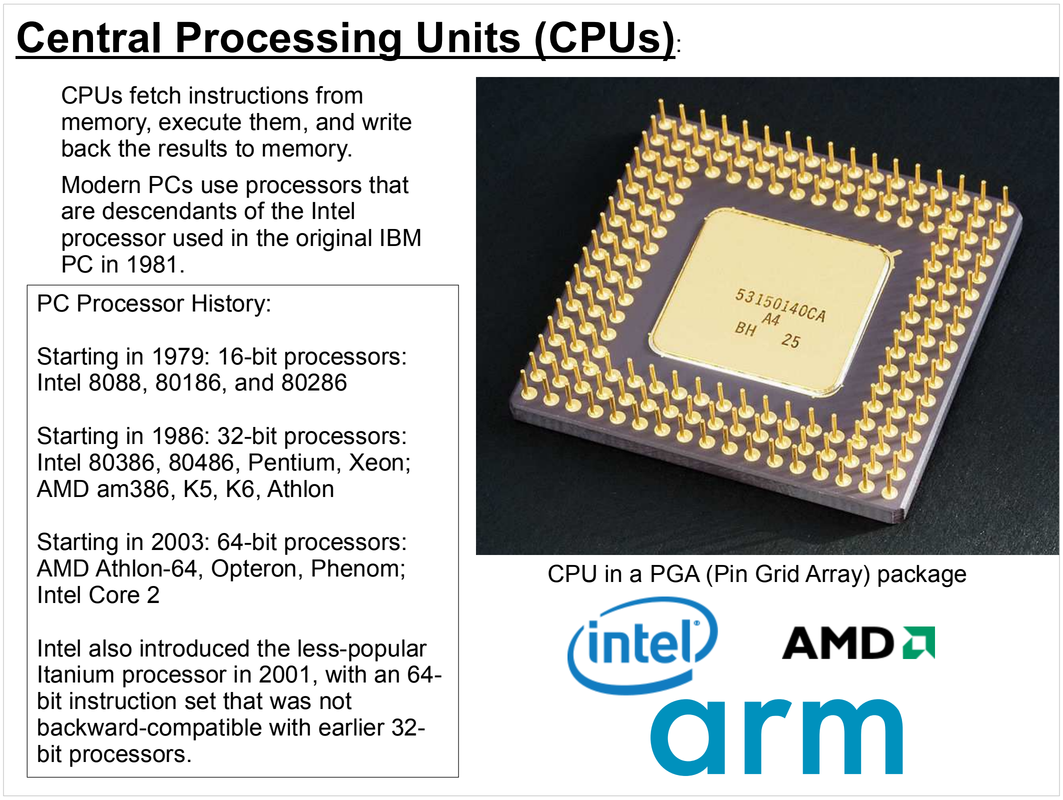
These bits are usually grouped together in sets of eight. A group of eight bits is called a “byte”.

1	0	1	1	0	1	1	0

Why eight bits? First, because eight is a power of two (2^3), making it convenient for binary (base-2) arithmetic. (Just as 10, 100 or 1000 are convenient in base-10.) Second, because the very popular early Intel CPUs used data in 8-bit chunks.

Some people claim that “bit” is a shortened form of “binary digit”, but I'm skeptical.

Half of a byte is sometimes referred to as a “nybble”.



Central Processing Units (CPUs):

CPUs fetch instructions from memory, execute them, and write back the results to memory.

Modern PCs use processors that are descendants of the Intel processor used in the original IBM PC in 1981.

PC Processor History:

Starting in 1979: 16-bit processors: Intel 8088, 80186, and 80286

Starting in 1986: 32-bit processors: Intel 80386, 80486, Pentium, Xeon; AMD am386, K5, K6, Athlon

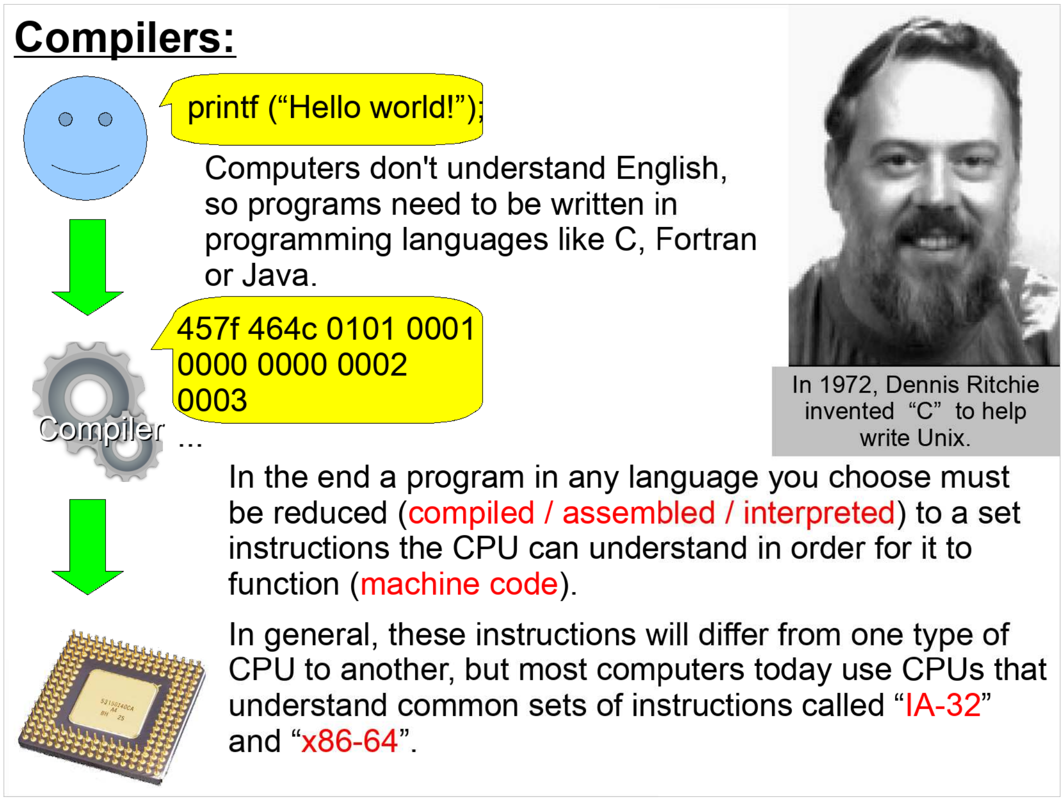
Starting in 2003: 64-bit processors: AMD Athlon-64, Opteron, Phenom; Intel Core 2

Intel also introduced the less-popular Itanium processor in 2001, with an 64-bit instruction set that was not backward-compatible with earlier 32-bit processors.

CPU in a PGA (Pin Grid Array) package

Now we'll start looking at CPUs.

Note that 32-bit processors can only deal with numbers up to 2^{32} (about 4 billion). Since many CPU operations involve the address of a chunk of data in the computer's memory, 32-bit processors have trouble dealing with any amount of memory larger than 4 billion bytes (4GB). This is one of the motivations for moving to 64-bit processors.



Compilers:

printf ("Hello world!");

Computers don't understand English, so programs need to be written in programming languages like C, Fortran or Java.

457f 464c 0101 0001
0000 0000 0002
0003

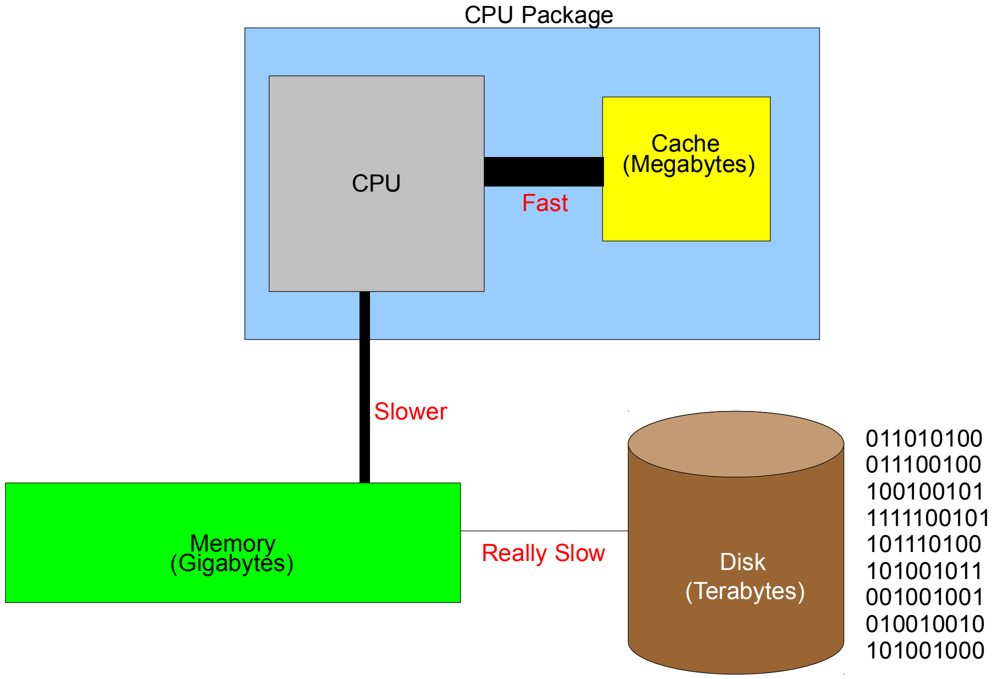
In 1972, Dennis Ritchie invented "C" to help write Unix.

In the end a program in any language you choose must be reduced (compiled / assembled / interpreted) to a set instructions the CPU can understand in order for it to function (machine code).

In general, these instructions will differ from one type of CPU to another, but most computers today use CPUs that understand common sets of instructions called "IA-32" and "x86-64".

The IA-32 instruction set is a descendant of the instruction set used by the processor in the orignal, 1981, IBM PC.

CPU Cache Memory:

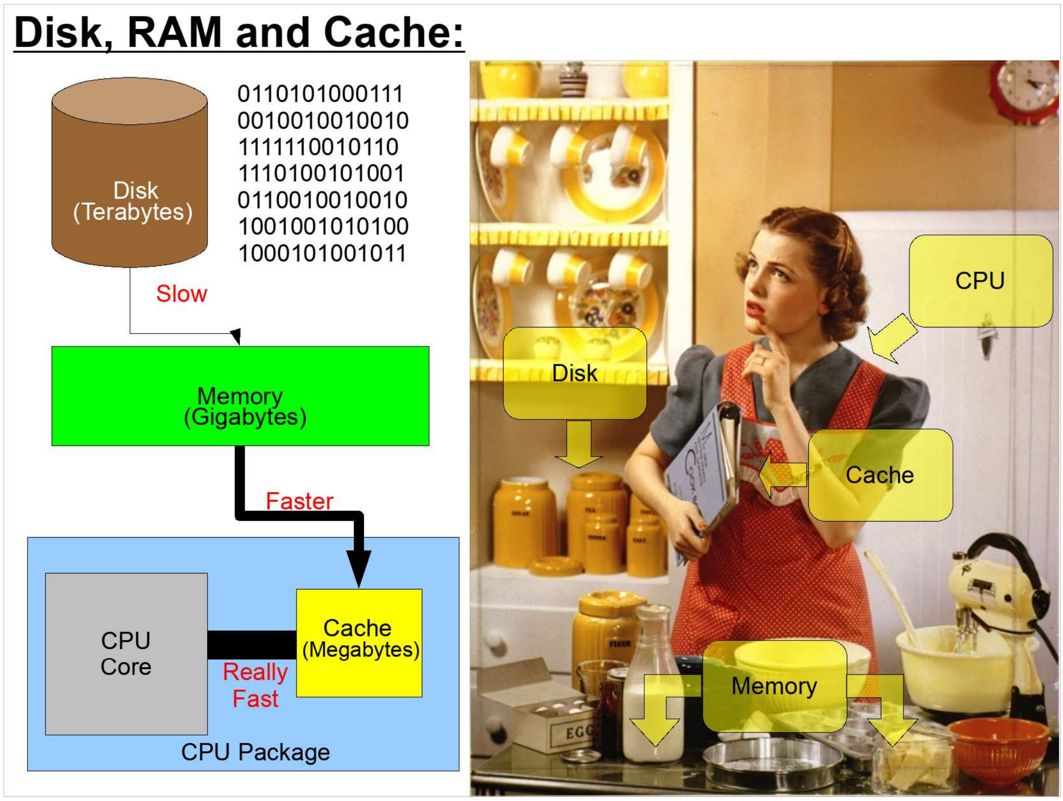


CPUs typically have a small amount of very fast memory called cache memory. Let's look at an example to see how cache memory is used.

Programs are typically stored on disks, which are big, but relatively slow. If the CPU had to fetch bits directly from the disk whenever it executed one of the program's instructions, the computer's performance would be terrible. Instead, parts of the program are stored temporarily in the computer's main memory, which is much smaller than the disk, but also much faster. The computer tries to make the best use of main memory by storing as many of the most important parts of the program there that it can. By important, here, we usually mean parts of the program that will be needed most often by the CPU. Sometimes the system will misjudge, and find that a needed part isn't currently in main memory. Then the CPU will wait while the necessary bits are fetched from the disk and stored in memory.

But even main memory isn't very fast compared with the CPU. Because of this, the CPU has a small amount of even faster memory called the cache. Just as data from disk is stored temporarily in main memory so it will be available quickly when needed, data from main memory are stored in the cache. When something that's needed isn't in the CPU's cache (this is called a “cache miss”), the system fetches it from main memory, if it's there, or all the way from the disk if necessary, while the CPU waits for the data.

I like to use a cooking analogy for all of this. The disk is like your kitchen cabinets, where you store all of your ingredients and utensils long-term. The main memory is like your kitchen counter. When you start cooking, you look through the cabinets and put the things you need onto the counter. If they won't all fit, you leave some in the cabinet and get them later, just putting the things you need first on the counter. Then you pick up the one or two things you need right now, for the task you're doing at the moment. Those things in your hands at the moment are in your cache memory. When you need an eggbeater instead of a spoon, you'll put down the spoon and pick up the eggbeater. That's a cache miss.



People are often confused by the terms “disk space”, “memory” (or “RAM”) and “cache”. I like to use a cooking analogy to explain the differences, and describe how computers compensate for the relatively low speed at which data can be read off of a disk.

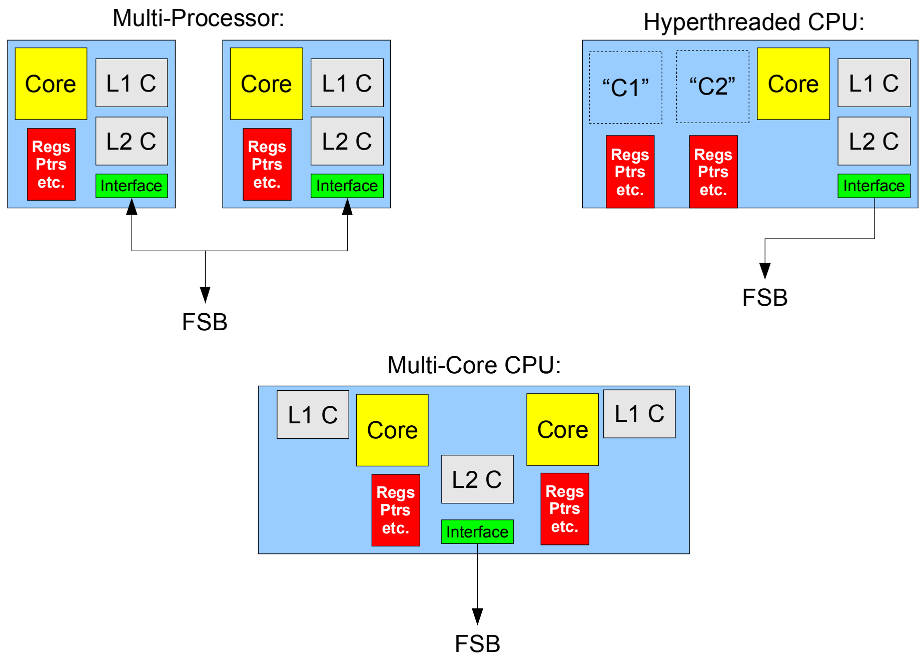
Since disk access is slow, computers try to predict which disk data you'll need next. Then, while the computer is digesting a bit of data you've just read from the disk, in the background it reads the data it thinks you might need next, and stores it in memory. The next time you request data from the disk, the computer first checks to see if the data is already stored in memory, and fetches it from there (much faster) if it is.

There's even a third level of storage, called “cache memory”, that's actually inside the CPU package itself, and can be accessed faster than RAM. Just like reading from the disk, the computer tries to predict which data from RAM you'll need next, and store it in the cache.

There's usually a trade-off between speed and size in computer storage:

Disk: large, slow, non-volatile
RAM: smaller, faster, volatile
Cache: tiny, really fast, volatile

Multi-Processor, Hyperthreading and Multi-Core:

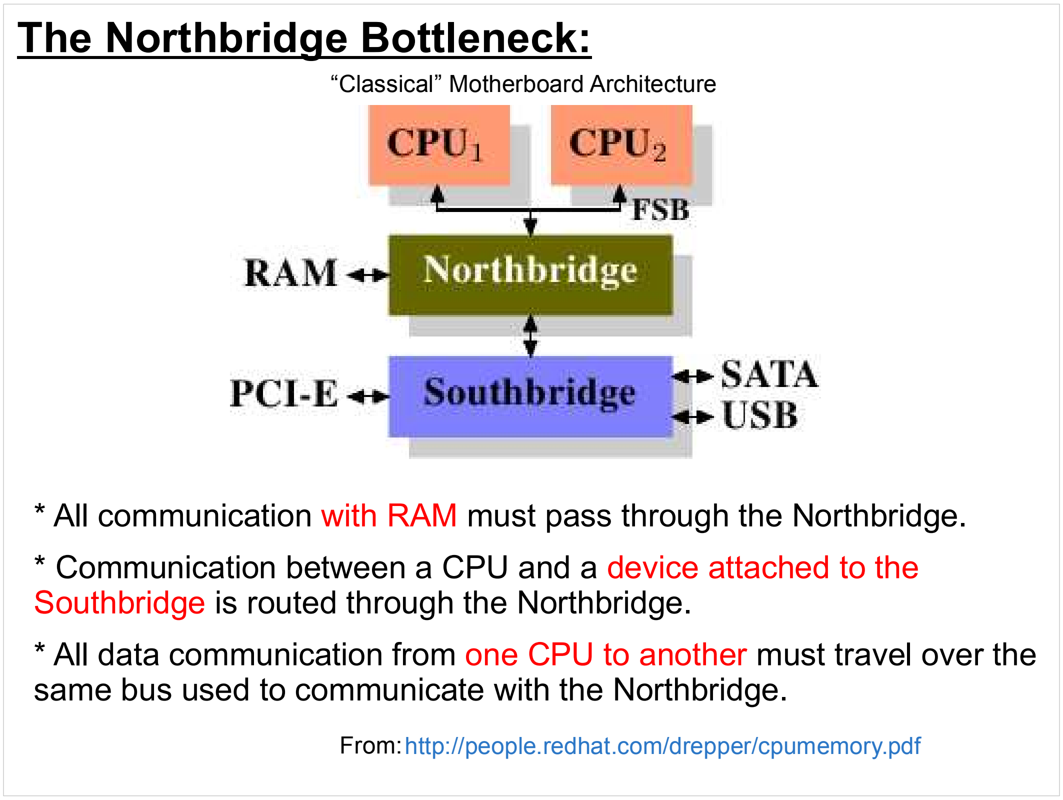


The figures above show several ways to make a computer faster by adding more CPU power. First, you could just add more CPUs. These are called multi-processor systems.

A few years ago, Intel developed a method for using a single CPU more efficiently, making it appear to be two CPUs. They did this by duplicating the registers and pointers that save a CPU's current state. These are essentially bookmarks that keep track of what the CPU is doing at a given time. When the CPU finds that it has to wait (perhaps for data from disk), it can simply hop over to the other set of registers and pointers, and go on with some other task until the first task can be continued. Intel calls this “hyperthreading”. With hyperthreading, a single processor appears to the rest of the system as though it were multiple processors.

Recently, it's become common to find multiple computing cores in a single CPU. A core on one of these multi-core CPUs may have its own cache, registers, pointers, etc., but share some resources with the other cores in the CPU.

Note that this figure shows two levels of cacheing: L1 and L2. In modern CPUs it's typical to have more than one level of cache. With two cache levels, data might flow from disk to main memory to the level 2 cache and then to the level 1 cache.



The Northbridge Bottleneck:

“Classical” Motherboard Architecture

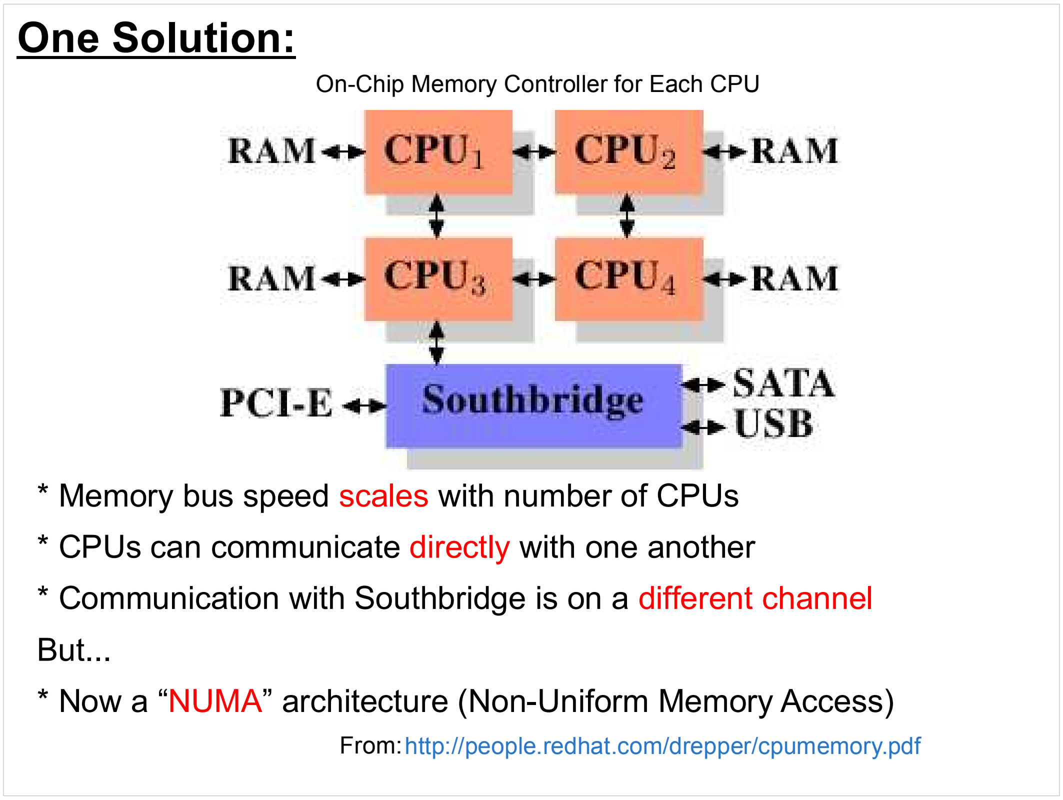
* All communication with RAM must pass through the Northbridge.

* Communication between a CPU and a device attached to the Southbridge is routed through the Northbridge.

* All data communication from one CPU to another must travel over the same bus used to communicate with the Northbridge.

From: http://people.redhat.com/drepper/cpumemory.pdf

The system of cacheing described just now works well, but it's not perfect. Cache misses are still normal, and we want the CPU to have to wait as little as possible. So, we need fast memory access. Several things conspire to slow down memory access in computers as they're currently designed.



One Solution:

On-Chip Memory Controller for Each CPU

* Memory bus speed scales with number of CPUs

* CPUs can communicate directly with one another

* Communication with Southbridge is on a different channel

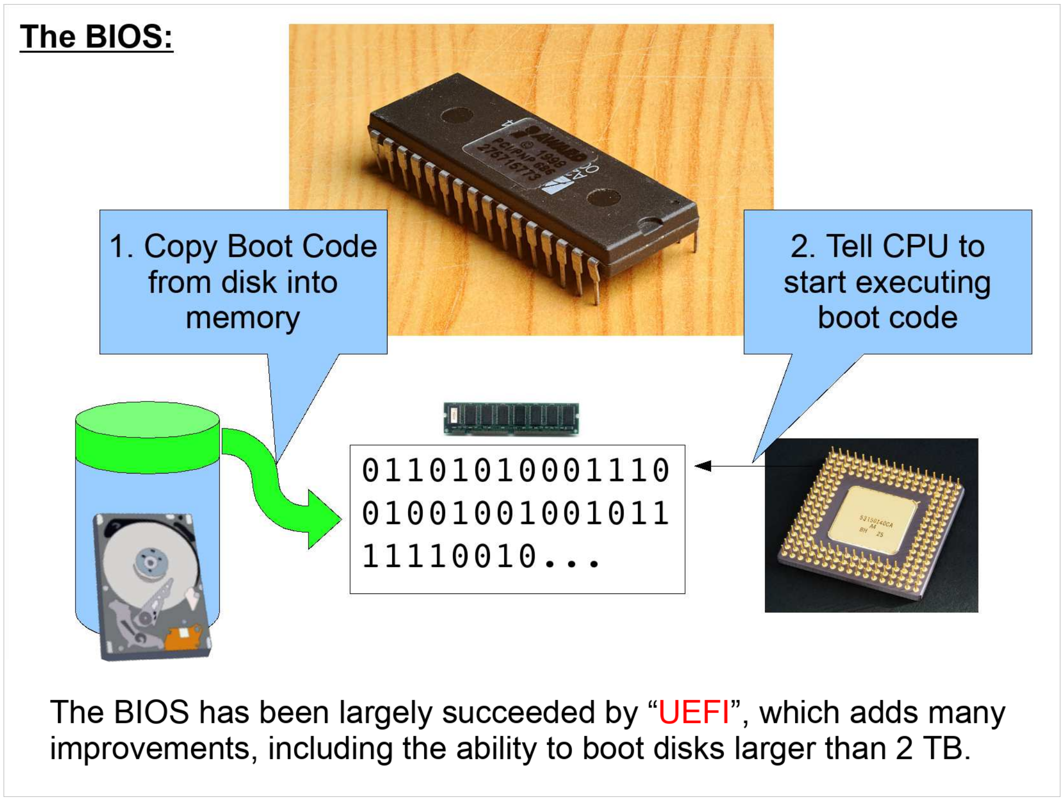
But...

* Now a "NUMA" architecture (Non-Uniform Memory Access)

From: http://people.redhat.com/drepper/cpumemory.pdf

Here's one way to speed things up. Vendors are already producing computers with designs like this.

Note that throughout this discussion of memory and cache I've had links to a paper by Ulrich Drepper. Please take a look at it sometime. It's a mind-bendingly thorough and detailed look at everything a programmer should know about computer memory.



The BIOS:

1. Copy Boot Code from disk into memory

2. Tell CPU to start executing boot code

01101010001110
01001001001011
11110010...

The BIOS has been largely succeeded by "UEFI", which adds many improvements, including the ability to boot disks larger than 2 TB.

The BIOS ("Basic I/O System") was once a much more important part of a computer than it is today. In current computers, the BIOS only has one job: To copy a little bit of data from the beginning of a hard disk (or other bootable device) into memory, and then tell the CPU to start reading that data.

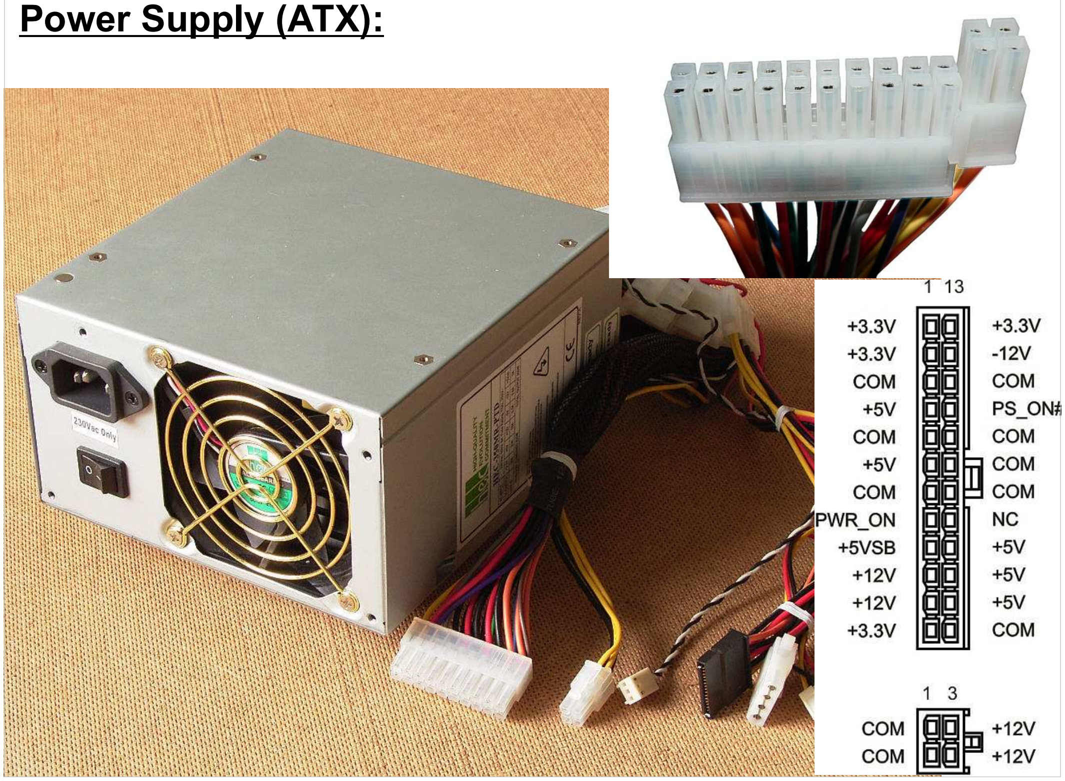
This is how the computer is "bootstrapped" ("booted") from doing nothing to running the operating system.

Here are some links to possibly interesting information.

http://www.tomshardware.com/reviews/bios-beginners,1126.html

http://en.wikipedia.org/wiki/Extensible_Firmware_Interface

Power Supply (ATX):



Finally, the lowly power supply. It's easy to replace if it dies (and it might).

The largest connector shown is for powering the motherboard itself. Older ATX motherboards required a 20-pin connector, newer ATX motherboards require 4 extra pins on a 24-pin connector. The best power supplies have connectors that can be split into 20-pin and 4-pin sections, for use in either old or new motherboards.

Among other power connectors shown are large 4-pin Molex connectors, which were the standard power connector for disks until recently. Newer SATA disks use a different, thinner power connector, found on all current power supplies.

One trick for diagnosing power supply problems: If you remove a power supply from a computer and plug it into the wall, you'll find that it doesn't turn on. This is because ATX motherboards have the ability to turn the power supply on or off. They do this by simply making a connection between two of the pins in the motherboard's power connector. Typically, these pins are connected to the connector's only green wire and an adjacent black wire. By shorting these together with a paper clip, you can cause the power supply to turn on. If this doesn't do it, the power supply is probably dead.

The End

Thanks!