assembly

We hear computers "think in binary". But what does that MEAN??

Your computer's memory (RAM):
01001001010010000000000000000000
0100

Some of this is cat pictures
Some of this is programs

Programs are binary:
01001011001010100

What does THAT mean?

Those are INSTRUCTIONS

Every CPU has an instruction set = (x86 or ARM) usually

Some instructions:
jmp mov add
xor push inc

Instructions are numbers

The inc instruction ("increment") on x86 is 1000000 or 0x40

Assemblers translate "human readable" assembly code into binary:

assembly code
mov $1, %rax
mov $1, %rdi
xor %rdi, %rdi

binar y
01001001
0....

Assembler

The instruction register contains an address in RAM.

CPU

IR
0x5884

I will go look in RAM there and run the code I find!
**The CAP Theorem**

From Martin Kleppmann's "A critique of the CAP theorem"

"C" stands for linearizable

in distributed systems, network partitions happen

???

hello?

computer

???

garbage collector!

(someone unplugged a cable!)

too much network traffic!

if you want to be consistent you can't always be available

panda

elephant

you're gonna have to wait for an answer

"CP systems"

consul

zookeeper

etcd

chubby

when they reply, you can believe them, but they don't always give you answers

"AP systems" available + partition tolerant

this doesn't mean very much.

always return "lol"

very carefully considered weaker consistency model

You can call both of these "AP"

CAP is a very simple theorem

I read the whole proof! It took 10 minutes + there's no math

CAP won't help you reason about most systems

I have a replicated database what can you tell me?

nothing!

CAP
copy on write

every time you start a new process on Linux, it does a `fork()` "clone" which copies the parent process.

the cloned process has EXACTLY the same memory.

3GB of RAM

old new

copying all the memory every time we fork would be slow and a waste of space.

the new process isn't even gonna use that memory most of the time.

so Linux lets them share RAM instead of copying.

Linux marks all the memory for both processes as read-only (in the page table).

I'm going to write to the shared memory!

no problem! I will just make a copy of that piece of memory.

UH OH that is not allowed!

Linux! PAGE_FAULT!

everyone is happy 🌟
directories + symlinks

What's a directory?

<table>
<thead>
<tr>
<th>filename</th>
<th>inode number</th>
</tr>
</thead>
<tbody>
<tr>
<td>awesome.jpg</td>
<td>279932</td>
</tr>
<tr>
<td>blah.txt</td>
<td>13227</td>
</tr>
<tr>
<td>cumberbatch</td>
<td>233333</td>
</tr>
</tbody>
</table>

I made a directory with 2,000,000 files.

It's so slow.

Listing your directory is gonna be REAL SLOW (a few seconds at least).

What's a symlink?

It's just a file with the name of another file in it?

$ readlink my-cool-link

/home/julia/long-complicated-file-name

OLD

On ext 2 even opening files in big directories is slow 😞

That's right! Ext 2 directories have no index so you have to SEARCH THE WHOLE THING 😞

Ext 2 is OLD though, ext3 is OK.
How does DNS work?

DNS servers translate names to IP addresses.

Where's cats.com?

172.23.9

This is called an A record.

Sometimes they tell you it's an alias (CNAME record).

Where's best.cats.com?

Same place as cats.com.

Most DNS queries get cached.

Where's cats.com?

8.8.8.8

Google DNS Server. Looks up cats.com for you and gives you an answer.

When an important DNS server dies.

Where's twitter.com?

8.8.8.8

Authoritative DNS server (dead).

I have no idea. It was at 172.23.9 but that DNS record expired and now the authoritative server is dead and AUGH
floating point

a double is 64 bits.
that means there are $2^{64}$ different doubles

going up to

$1.8 \times 10^{308}$

there are $2^{52}$ numbers between 1 and 2

$1 + \frac{1}{2^{52}}, 1 + \frac{2}{2^{52}}, \ldots$

$2^{51}$ numbers between 2 and 4

$2^{50}$ between 4 and 8 etcetera.

some double arithmetic

$2^{52} + 0.2 = 2^{52}$ (the next number after $2^{52}$ is $2^{52} + 1$)

$1 + \frac{1}{2^{53}} = 1$ (the next number after $1$ is $1 + \frac{1}{2^{52}}$)

$3 \times 10^{100} = \text{infinity} \leftarrow \text{infinity is a double}
\text{infinity - infinity = \text{nan} (not a number)}$

Javascript only has doubles (and \text{Lua}?)

that means after $2^{53}$ you don't have every integer!

printing doubles is nontrivial

the shortest version of 25.64853898042e8
is 2.564854e9

calculating the shortest representation takes time!
IPv6

What will we DO?!!

IPv4 addresses are 32 bits

IPv6 addresses 128 bits

2001:0db8:85a3:0000:0000:8a2e:0370:27a9

that's cool! I totally understand IPv6 because this is 2016

Windows 2000 had IPv6 support, operating systems: SO READY

adoption:

it's happening!!!

Google says 30% of American traffic they see is using IPv6

people were putting it off but we're REALLY RUNNING OUT of IPv4 addresses so now they have no choice
the "OSI model" for networking

I don't always find it useful but it's good to know what "layer 4" means.

what does "this is an L4 proxy" mean?

If a load balancer is labelled "L7" it usually means it looks at the Host: header inside your HTTP packets.

I only know about IP addresses! I don't even know what a port is let alone what the packet says.

LAYERS

1: electrical engineering stuff, wires, frequencies, wifi
2: Ethernet protocol + others
3: IP (IP addresses)
4: TCP + UDP (ports)
5+6: nobody ever talks about these
7: HTTP and friends
What's a MAC address?

Every computer on the internet has a network card.

Hello! You can call me 0a:58:ff:ea:05:97.

MAC address

When you make HTTP requests with Ethernet/wifi, every packet gets sent to a MAC address.

Here is a cat for 0a:58:...

Yay!

0a:58

Wait, how do I know someone else on the same network isn't reading all my packets?

You don't! That's one reason we use HTTPS + secure wifi networks.

Your router has a table that maps IP addresses to MAC addresses.

A message for 192.0.2.77?

I will send that to 0a:58:ff:ea:05:97!

(read about ARP for more)

More at: drawings.jvns.ca!
memory allocation

at any given time, your program has a fixed amount of memory:

- used
- free

587 MB

this was used but then got freed

and it can ask the OS for more memory:

- Google Chrome

now I have 1.8 GB of memory! yay!

your allocator tries to fill in unused pieces when you ask for memory:

- can I have 512 bytes of memory?

  - YES
  - malloc

you can invent your own strategy to allocate memory:

- glibc malloc's algorithm is dumb, I'm going to do my own thing

this is sort of normal to do if you care a LOT about performance.

alternatives to libc malloc:

- jemalloc
- tcmalloc

Facebook

Google
man pages = awesome
(sometimes quality may vary)

1. programs
   - $man grep
   - $man ls

2. system calls
   - $man sendfile

3. C functions
   - $man 3 printf
   - $man fopen

4. devices
   - $man null
   - For /dev/null docs

5. file formats
   - $man sudoers
   - For /etc/sudoers

6. games
   - Not very useful
   - man sl is good if you have sl though

7. miscellaneous
   - $man 7 pipe
   - $man 7 symlink
   - (these are cool!)

8. sysadmin programs
   - $man apt
   - $man chroot

I found out I can get documentation for programs (like grep) with man grep!

but that's not all!!
lots of other things have man pages too!

man pages are split up into 8 sections
1 2 3 4 5 6 7 8

/usr/share/man/man5
has section 5 on my machine.

GREAT

Julia Evans
@b0rk
mesos manages resources

master

agents

We have 200 CPUs + 800 GB of RAM. What should we do?

agents run "tasks"

running on

agent #99

needs 2 GB of RAM + 3 CPUs

program ← state: running

the Mesos master keeps track of EVERY running task

... dude there are THOUSANDS of these things. I got it though.

frameworks ask the Mesos master to run tasks. There are LOTS.

Marathon (HTTP services)
Chronos (cron-like jobs)
Jenkins
Spark
Hadoop
ElasticSearch
Cassandra

you can split your Mesos cluster between several frameworks

half for Hadoop, half for web services?

Mesos doesn't know much about tasks

task

OK what it's doing

that's a HTTP service running on port 9923

mesos

Marathon
Sometimes you're running code on 2 CPUs at the same time:

- CPU 1: $x = 2$
- CPU 2: $x = 3$

Sometimes 2 threads want to change the same thing:
- Program 1 writes "a"
- Program 2 writes "u"

Array:
- Current value: "hello"

A mutex keeps track of whether something is in use:
- Program 1's turn
- Mutex available

When you're done, you tell the mutex it's available:
- Program 1: I'm done!
- Program 2: Yay!

There's lots more, but we're outta space:
- Semaphores
- Futex
- Compare and swap
- Atomic instructions
networking concepts

hey I want to understand all the networking stuff that happens when I go to google.com!

YES that is awesome. there are a lot of concepts but you can totally learn them all!

(knows many networking concepts now)

protocols

DNS
SSL/TLS
IP
TCP
UDP
ICMP (ping)
ARP
BGP
ethernet

other concepts

socket
packet
port
IP address
nameserver
NAT
router
TTL
checksum

it's a lot to learn but it's totally possible to learn how it all fits together to get you pictures of cats 😊
anatomy of a packet

When you get a webpage like Facebook, it comes into your computer in many small packets.

Let's see what those look like!

Packets are split into a few sections (or "headers")

"physical layer": this gets changed constantly as your packet moves between computers.

in charge of getting your packet to the right server (like an address on an envelope)

in charge of preventing data corruption and helping you retry lost packets

video streaming uses UDP instead. UDP does not try to be reliable.

the actual data you're trying to send is
Page Table (in 32-bit memory)

- Every process has its own memory space.
  - Address 0x40000000 at that address says "cat" for me it says "dog".
  - Process 1
  - Process 2

- Each address maps to a 'real' address in physical RAM.
  - Process 1 -> 0x28ea4000
  - Process 2
  - 0x3942f000

- Processes have a "page table" in RAM that stores all their mappings.
  - 0x12345000 -> 0xae9250
  - 0x23f49000 -> 0x123450

- The mappings are usually 4kB blocks (4kB is the normal size of a "page").

- Every memory access uses the page table.
  - I need to access 0x40000000.
  - CPU
  - The page table says the real address is 0x9923456.
  - Sort of.

- When you switch processes...
  - Kernel: Here, use this page table instead now.
  - Okay thanks!

- Some pages don’t map to a physical RAM address.
  - I’m gonna access 0x00040000
  - EEPROM!
  - BAD ADDRESS!
  - CPU
  - => Segmentation fault =

Unix permissions

3 kinds of things you can do to a file:
- read
- write
- execute

```bash
$ ls -l
rw- rw- r--
```

- bork can do this
- staff can do this
- ANYONE can do this

```bash
$ ls -l awesome.png

rw- rw- r--
```

- bork
- staff
- ANYONE

```
$ ls -l /bin/ping

rw- r-x r-x
```

- setuid flag
- This means ping always runs as root (who owns it), no matter who started ping

```bash
755 business?
7 → rwx
6 → rwx
5 → r-x
4 → r--
```

- it's binary?
- 7 → 111 → r-x
- 755 means
  - rwx
  - r-x
  - r-x

more weird permissions things

- setgid
- sticky bit

but I ran out of space
Sometimes you want to send the output of one process to the input of another:

```
$ ls | wc -l
53
```

A pipe is a pair of 2 magical file descriptors and

```
ls  wc
stdin  stdout
```

When `ls` does write(, “hi”) `wc` can read it!
```
read( )
-> “hi”
```

Pipe buffers

`ls` I'm gonna write a bajillion bytes to

```
ls
```

Uh no if my buffer is full you have to wait

```
ls
```

What if your target process dies?

```
ls  wc
```

`ls` gets sent SIGPIPE if gets closed (ls usually dies)

You can pipe SO MANY things together

```
$ a b c d e
```

Pairs of pipes
asking good questions

find a good time
hey can I ask you about database performance for 20 minutes?
yeah! can we do it after lunch?
yeah!

state what you know
so, I know when the database gets a lot of writes, the hard drive can't keep up.
that's right! I don't think that was our problem though, look at this...

ask factual questions
does this database take out a lock when it does writes?
yes! here are the docs you should read if you want to know more! They're good.

choose who to ask
probably a better choice, has a good shot at answering your questions faster.
your coworker with a bit more experience than you

do some research
so I found out that creating database indexes takes time and I have questions about how that affects performance...
great

profit
now I know a lot more
I really helped! That was a great use of time
CPU scheduling

Once upon a time...

I want to run a program!

Sorry I can only do 1 program at a time.

Every CPU core can only run 1 program at a time.

OK time to stop it's Jimmy's turn to use the CPU now.

Operating System

Every program gets a few ms at a time.

Steps when we switch the running process:

→ Save:
  * Registers
  * Stack pointer
  * Which CPU instruction to start at next time

→ Set up memory for new process

→ Load new registers and stuff

All this takes time (2 microseconds?).

It's ok to do but you don't want to be switching processes constantly.

You don't use the CPU when you're waiting

Hey I'm waiting for a network response.

Cool! I'll run other stuff until that comes back.

OS
The Stack

(in a C program)

Your program has

- local variables
  ```
  int x = 2;
  ```
- a function to return to
  ```
  void parent() {
    doThing();
  }
  ```
- function arguments
  ```
  makeCat(name, fluffiness)
  ```

These all live in a part of memory called the stack.

Example program:

```c
int fun () {
  int x = 2;
  doThing(2);
  int y = 4;  //
  }

void doThing(b) {
  int x = b+1;
  int z = 4;
  return;
}
```

The stack at:

```
locals
- 3 (x)
- 4 (z)
- address for 2 (b)
```

Return address:
```
2 (x)
```

There's a limit to how big your stack can get! Exceed it and you get a stack overflow.
What's a thread?

A process can have lots of threads:

<table>
<thead>
<tr>
<th>process id</th>
<th>thread id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1888</td>
<td>1888</td>
</tr>
<tr>
<td>1888</td>
<td>1892</td>
</tr>
<tr>
<td>1888</td>
<td>1893</td>
</tr>
<tr>
<td>1888</td>
<td>2007</td>
</tr>
</tbody>
</table>

Threads share memory:

- I'm going to write "3" to address 2977886
- I can overwrite that if I want!

But they can run different code:

- Thread 1: I'm doing a calculation!
- Thread 2: I'm waiting for a network request to finish!

Sharing memory can cause problems: "race conditions"

- Thread 1: I'm going to add 1 to that number!
- Thread 2: I'm going to add 1 to that number!

Result: 24 ← wrong (should be 25)

If you have 8 CPU cores, you can run code for 8 threads at the same time:

| 1  |
| 5  |
| 2  |
| 6  |
| 3  |
| 7  |
| 4  |
| 8  |

SO BUSY 😞 😞
User space vs. kernel space

The Linux kernel has millions of lines of code:
- read/write files
- decide which programs get to use the CPU
- make the keyboard work

When Linux kernel code runs, that's called kernel space.
When your program runs, that's user space.

Your program switches back and forth:
```
str = "my string"
x = x + 2
file.write(str) ↩️ switch to kernel space ↩️
y = x + 4
str = str * y ↩️ and we're back to user space!
```

Timing your process:
```
$ time find /home
0.15 user 0.73 system
```

Time spent in your process:
```
```
Time spent by the kernel doing work for your process:
```
```
VIRTUAL MEMORY

I'd like to read from this address in memory!

it's actually in RAM

"translation lookaside buffer"

translates virtual addresses to real addresses

TLB → RAM

OH NO. that address is actually on disk!

PAGE FAULT!

HARD DRIVE

huh, why is that memory address actually on the hard drive?

reason ①

mmap

lets you map a bunch of stuff on disk into memory. None of it will actually get read from disk until you access the memory.

reason ②

swap

if you run out of memory, it gets saved to disk and your computer gets SUPER SLOW 😞